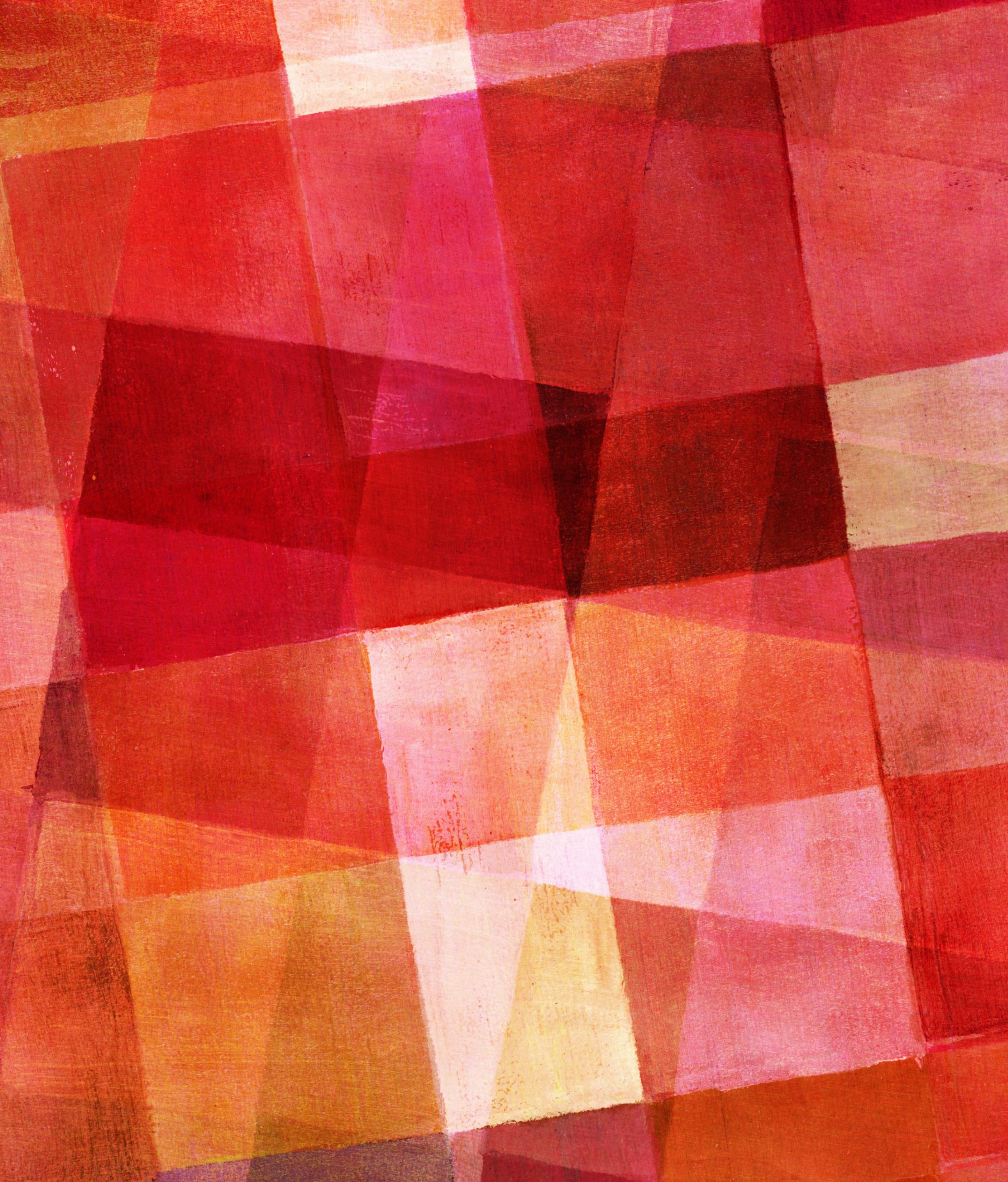




FUNDAMENTOS DE SQL



FUNDAMENTOS DE SQL

ENUNCIADO

FUNDAMENTOS DE SQL

ENUNCIADO

ENUNCIADO

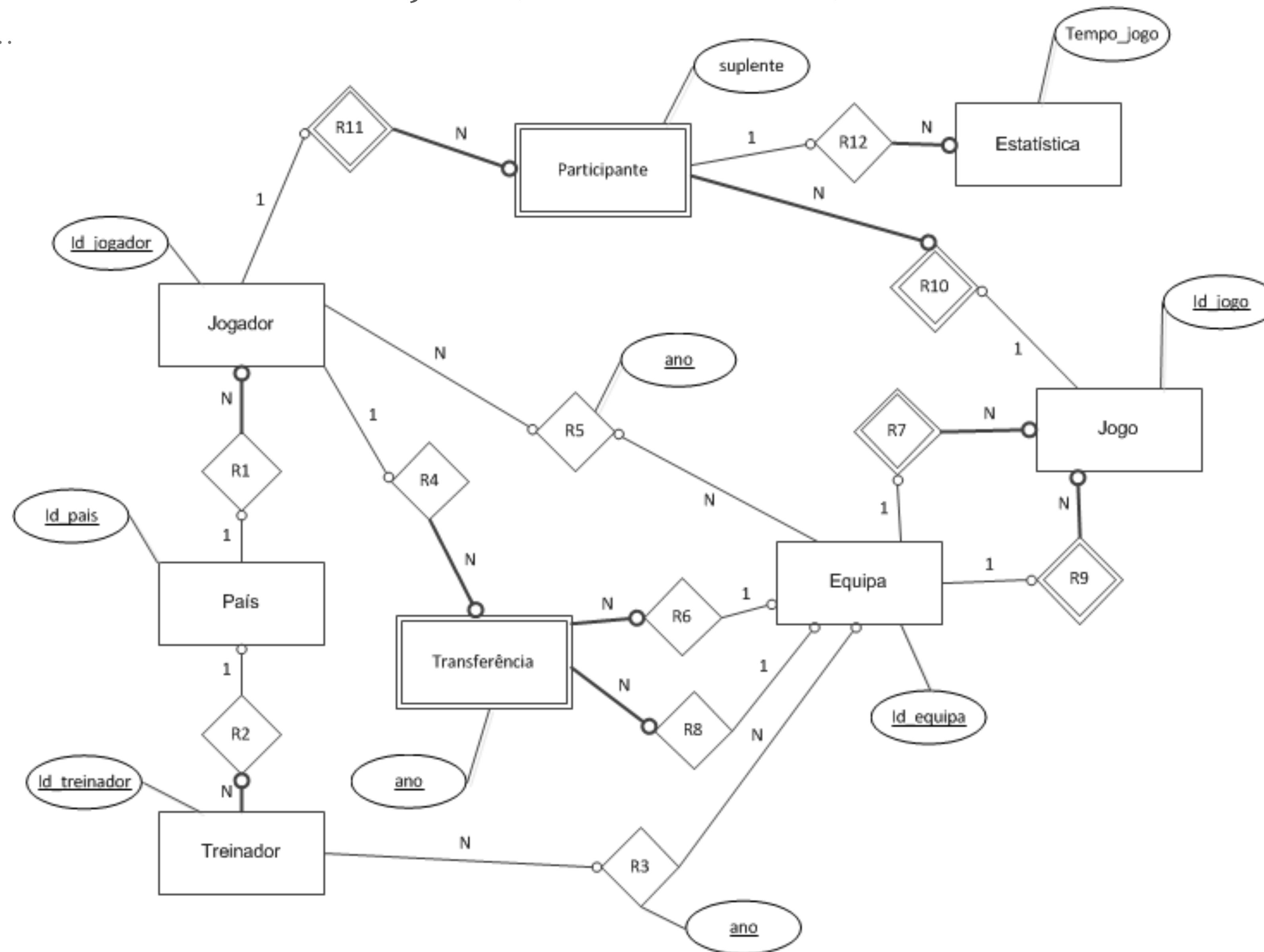
Uma associação de futebol pretende desenvolver uma base de dados onde serão registados todos os factos relevantes relativos às competições geridas por esta. A base de dados deverá gerir as entidades tradicionais do mundo do futebol, a saber: jogadores, treinadores, equipas, nacionalidades, jogos realizados, participantes nos jogos e ocorrências verificadas (golos, cartões amarelos, cartões vermelhos, etc).

Como as informações, nunca são removidas, dados estatísticos poderão ser retirados da base de dados de modo a obter diversos tipos de relatórios como por exemplo: o tempo total de jogo de um jogador.

Para definir a base de dados, foi considerado o seguinte:

- *O contrato de um treinador ou um jogador, com uma equipa, tem a duração mínima de 1 ano;*
- *Para facilitar, só são registados jogos com equipas do mesmo país;*
- *O tempo de jogo é registado ao segundo;*

MODELO ENTIDADE-ASSOCIAÇÃO (SIMPLIFICADO)



LEGENDA – RELACIONAMENTOS E ENTIDADES

Relacionamentos:

Relacionamento	Relacionamento	Relacionamento
R1: Naturalidade	R6: Transferido_de	R11: Regista
R2: Naturalidade	R7: Visitada	R12: Tem
R3: Assina Contrato	R8: Transferido_para	
R4: Transferido_para	R9: Visitante	
R5: Assina Contrato	R10: Composto_por	

Entidades:

Pais – País de origem

Jogador – Jogadores

Transferência – regista as transferências de jogadores entre equipas

Treinador – Treinadores

Participante – Jogadores titulares e suplentes de um jogo

Estatistica – regista todas as ocorrências de um jogador num jogo

Equipa – Equipas das competições

Jogo – Jogo

LEGENDA – ATRIBUTOS DAS ENTIDADES/RELACIONAMENTOS

País	Jogador	Jogo	Equipa
Id_Pais: Código do País	Id_jogador: Código do Jogador	Id_jogo: Código do Jogo	Id_equipa: Código da equipa
Nome: Nome do País	Nome: Nome do Jogador	Data: data do jogo	Nome: nome da equipa
Transferência	Data_nasc: Data de nascimento	Ano: ano da competição	Criação: ano da criação do clube
Valor: valor da transferência	Salario: vencimento mensal do jogador	Golos_casa: golos da equipa da casa	NumSoc: número de sócios
Treinador	R5	Golos_fora: golos da equipa visitante	Estádio: nome do estádio
Id_treinador: Código do treinador	Ano: ano do contrato do jogador	Assistência: número de espectadores	Lotação: capacidade
Nome: Nome do treinador	R3	Jornada: número da jornada	
Estatística	Ano: ano do contrato do treinador		
Tempo_jogo: tempo de jogo do jogador			
Ocorrência: tipo de ocorrência			

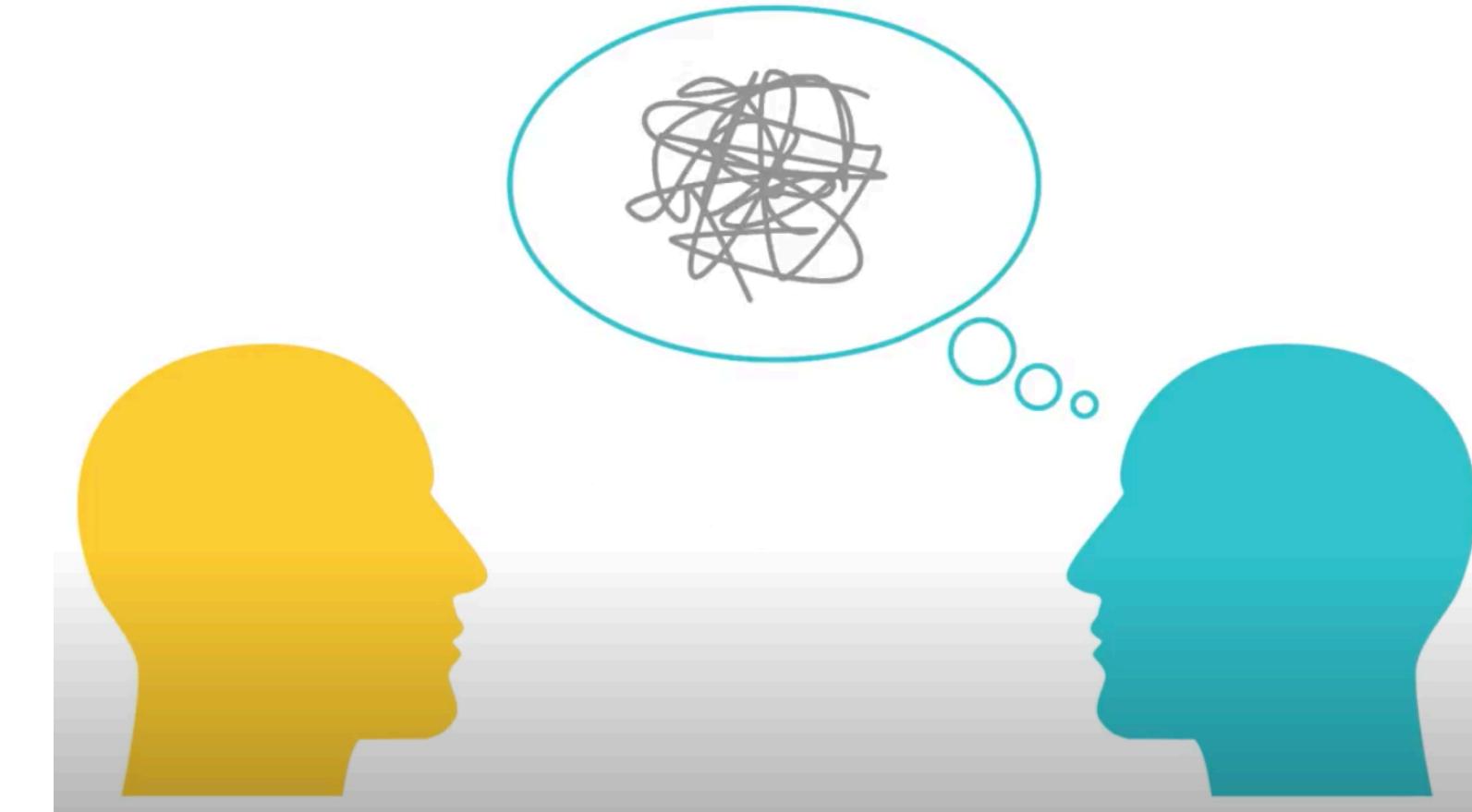
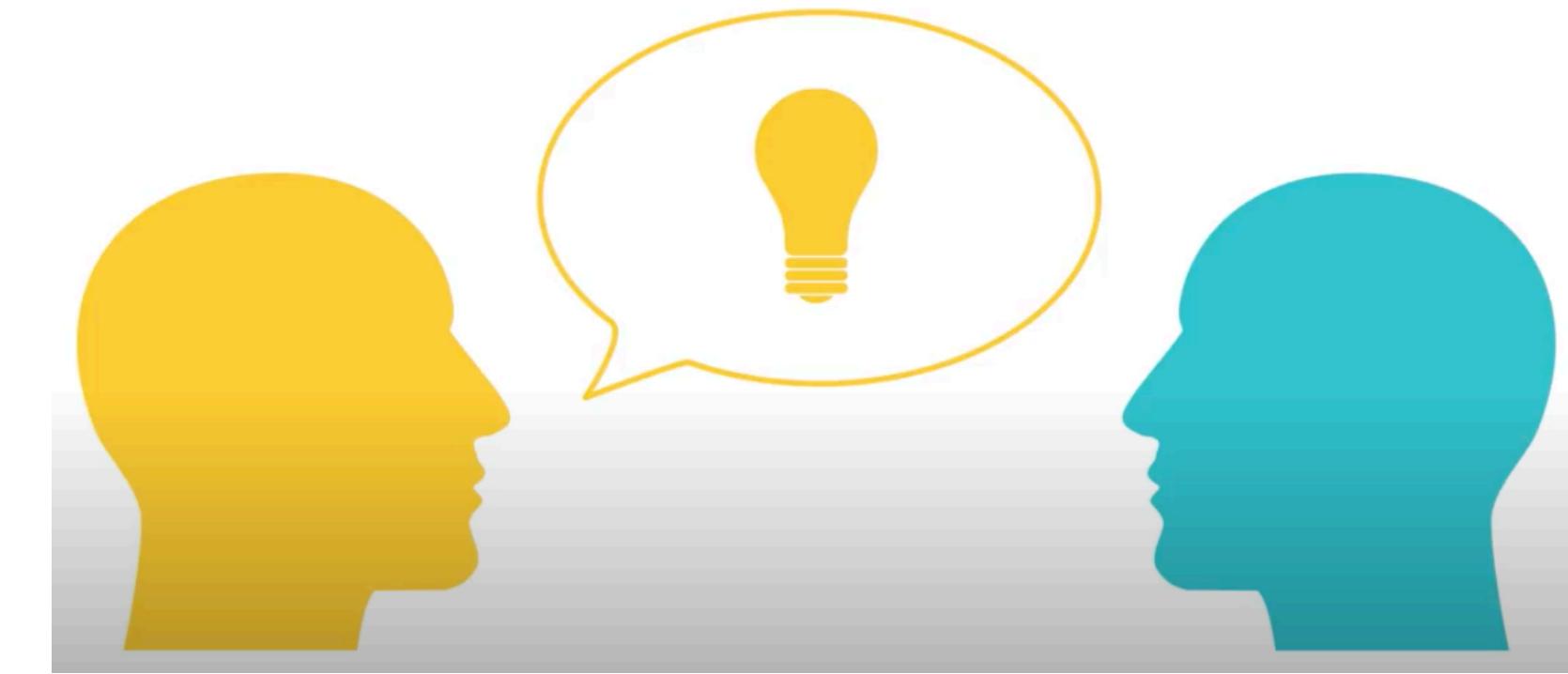
FUNDAMENTOS DE SQL

Diagramas de caso de uso UML

DESENVOLVIMENTO COLABORATIVO - UML

Contexto Geral:

- A pessoa A teve uma ideia e tenta partilhar com a pessoa B;
- Contudo a pessoa B não consegue perceber a mesma ideia;

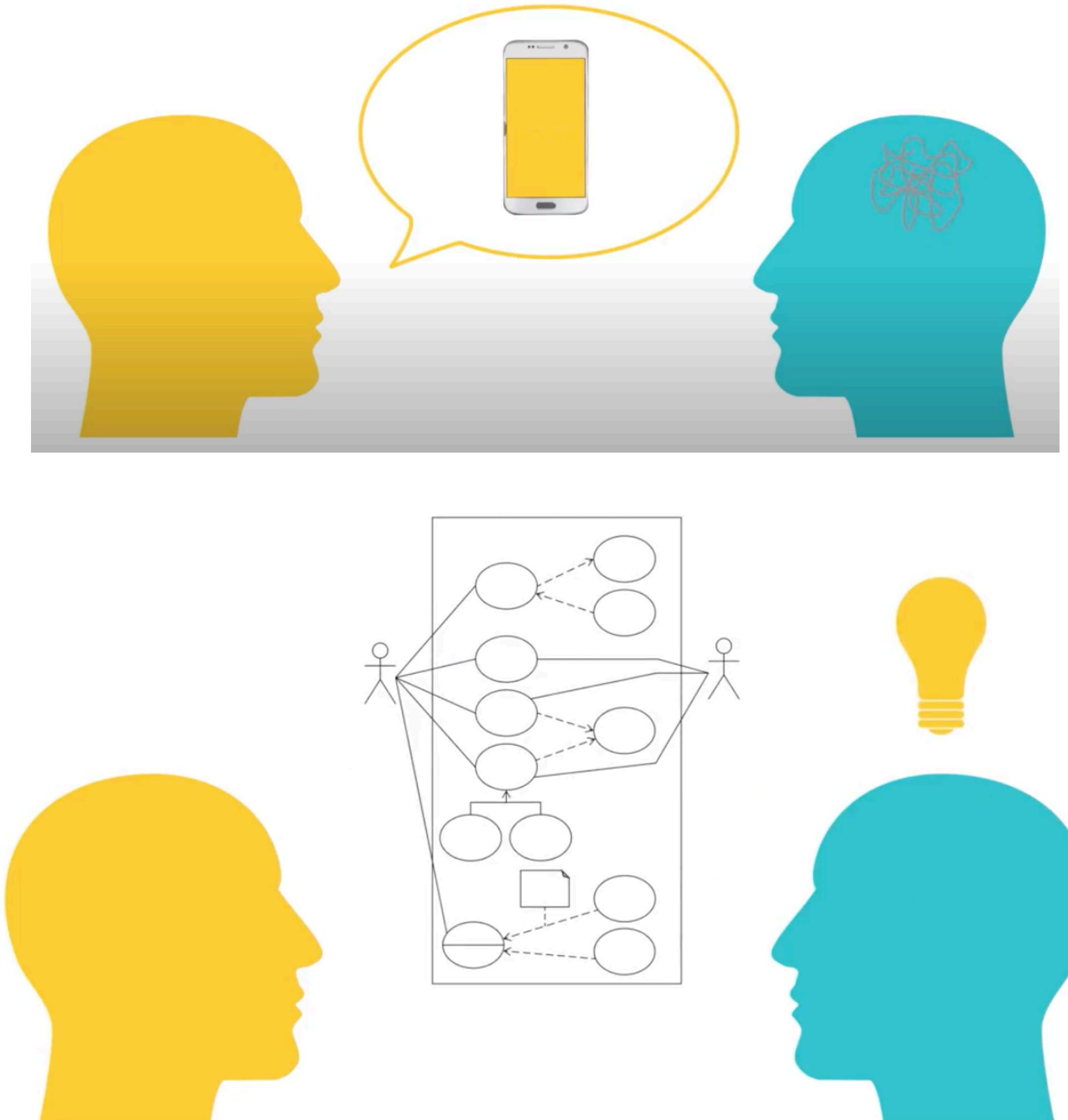


DESENVOLVIMENTO COLABORATIVO - UML

Contexto empresarial:

- A pessoa A explica o mecanismo de uma aplicação de software;
- A pessoa B não consegue perceber o funcionamento da mesma. Mesmo que a mesma seja extremamente simples;

Neste contexto recorremos aos use cases UML que facilita bastante o funcionamento da aplicação.

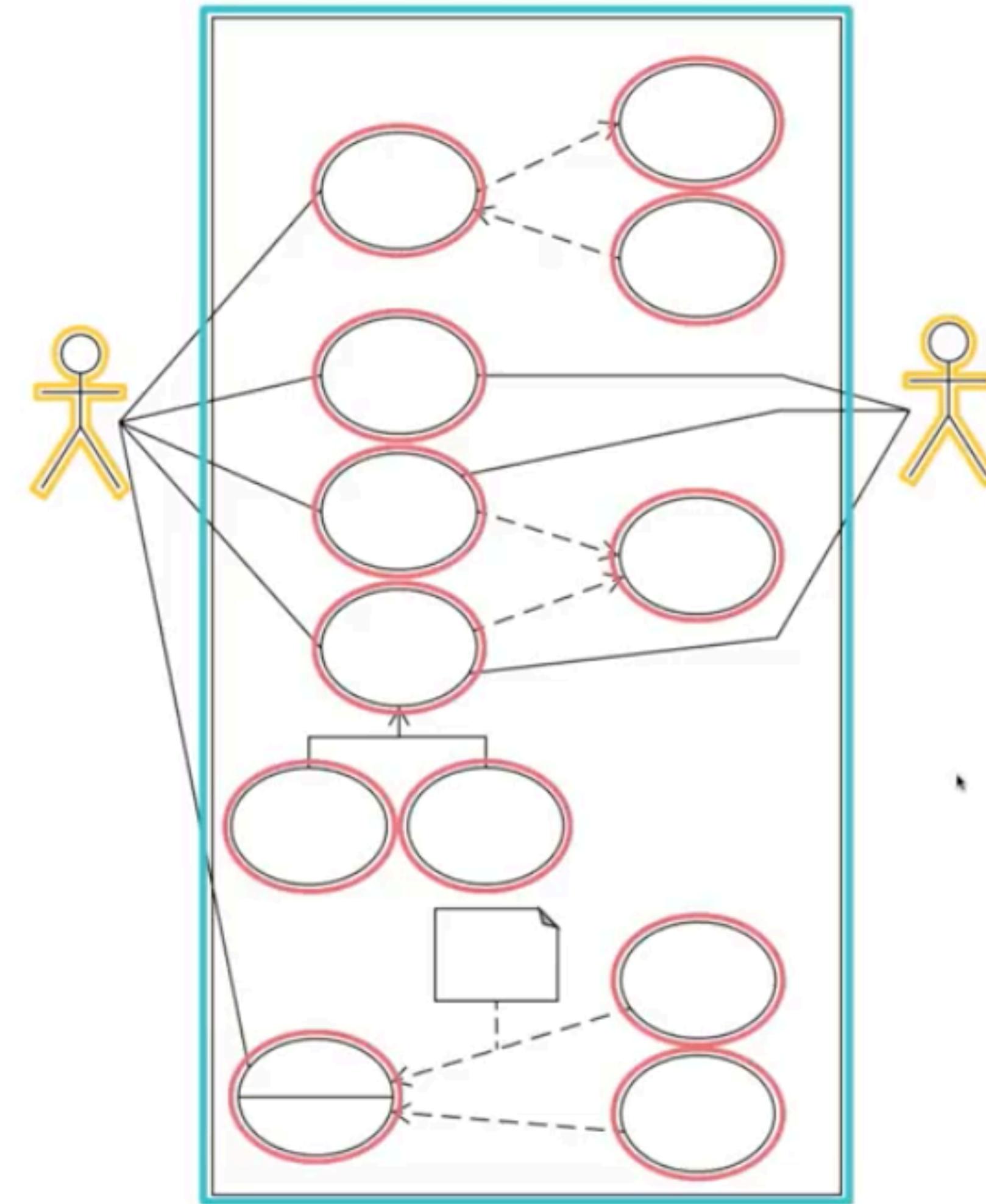


DESENVOLVIMENTO COLABORATIVO - UML

Processo que comprehende as seguintes características:

- Sistema ou aplicação (cor azul);
- Pessoas ou organizações que interagem com o sistema (cor amarela);
- Flow básico que demonstra o que o sistema trata;

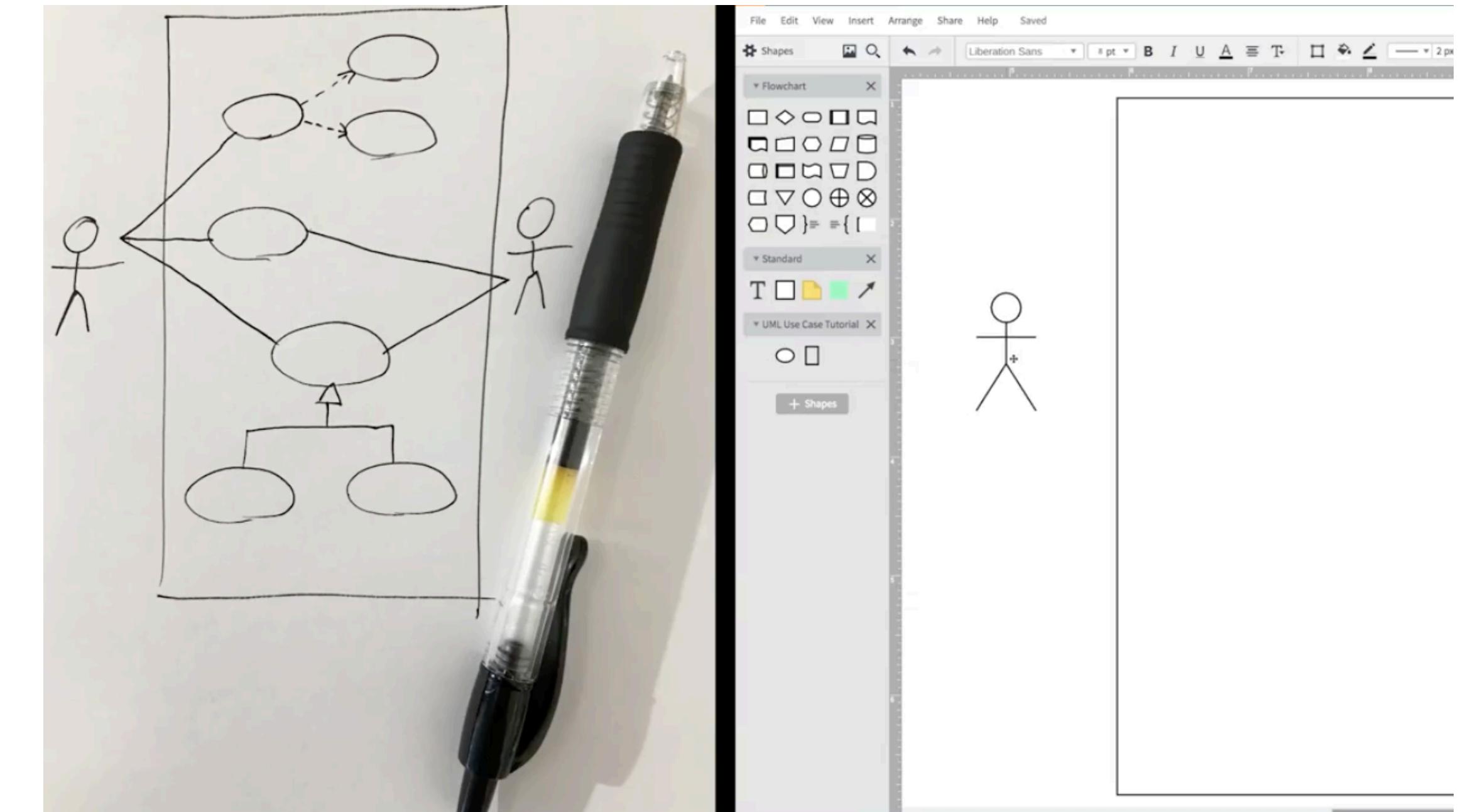
Diagrama de alto nível, pode não demonstrar grande detalhe mas é uma excelente forma de resolver situações como as anteriores.



DESENVOLVIMENTO COLABORATIVO - UML

COMO CRIAR UM USE CASE UML?

- Passível de ser criado com o auxilio de uma caneta e papel;
- Plataforma (grátis) como alternativa à anterior que permite a criação deste processo, nomeadamente “LucidChart”;

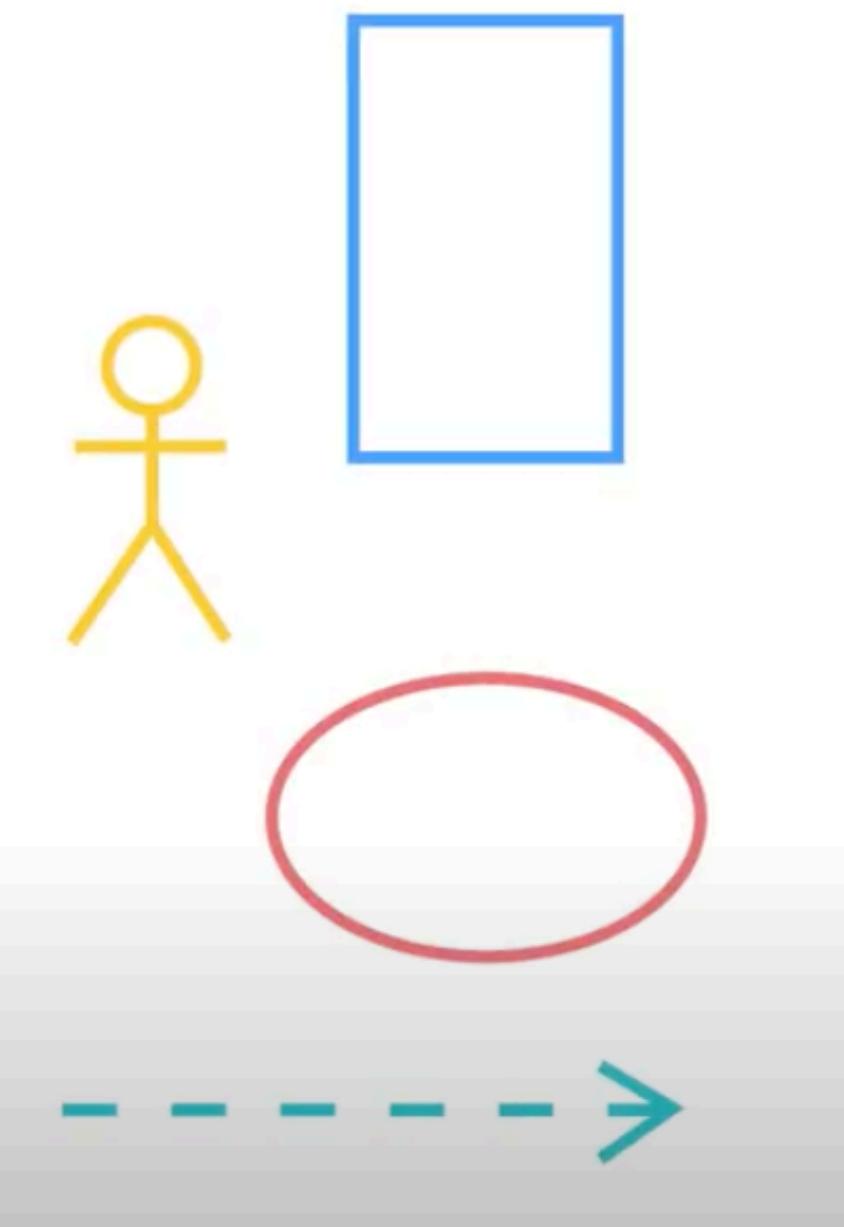


DESENVOLVIMENTO COLABORATIVO - UML

Existem 4 características essenciais na construção dos cases UML:

- **System;**
- **Actors;**
- **Use Cases;**
- **Relationships.**

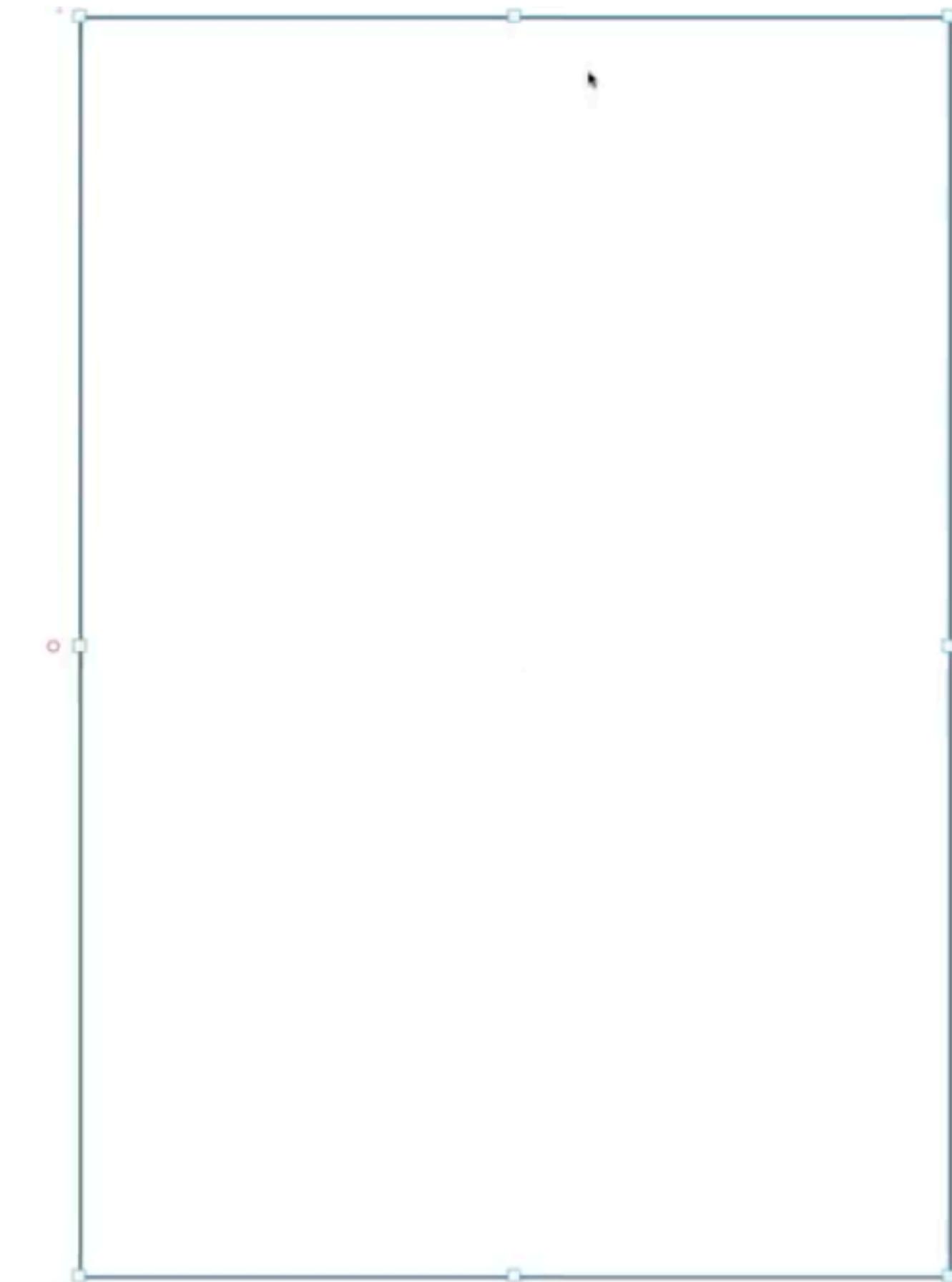
Systems
Actors
Use Cases
Relationships



DESENVOLVIMENTO COLABORATIVO - UML

SYSTEM (sistema) - Algo a ser desenvolvido.

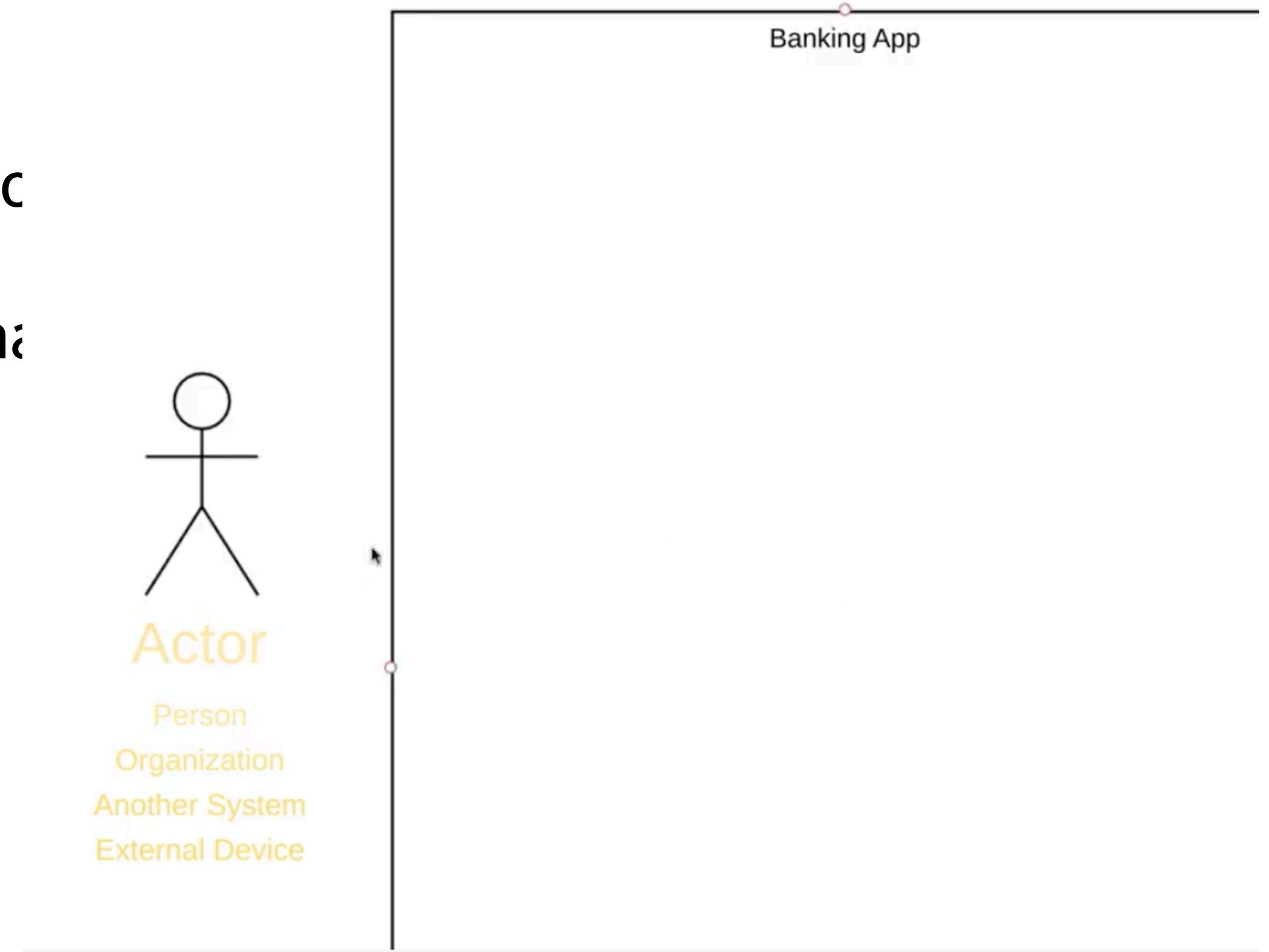
- Poderá ser:
 - WebSite;
 - Componente de software;
 - Processo de negócio;
 - Aplicação;
 - Entre outras.
- Representa-se por ser o rectângulo onde vai conter toda a informação;
- Deverá ser preenchido o nome do projecto no topo do rectângulo.
- Este rectângulo vai conter toda a informação inerente ao sistema.
- Tudo o que esteja fora deste rectângulo não entra no funcionamento inerente ao sistema contudo pode ser influenciado.



DESENVOLVIMENTO COLABORATIVO - UML

ACTOR (actor) - Elemento que influencia o sistema.

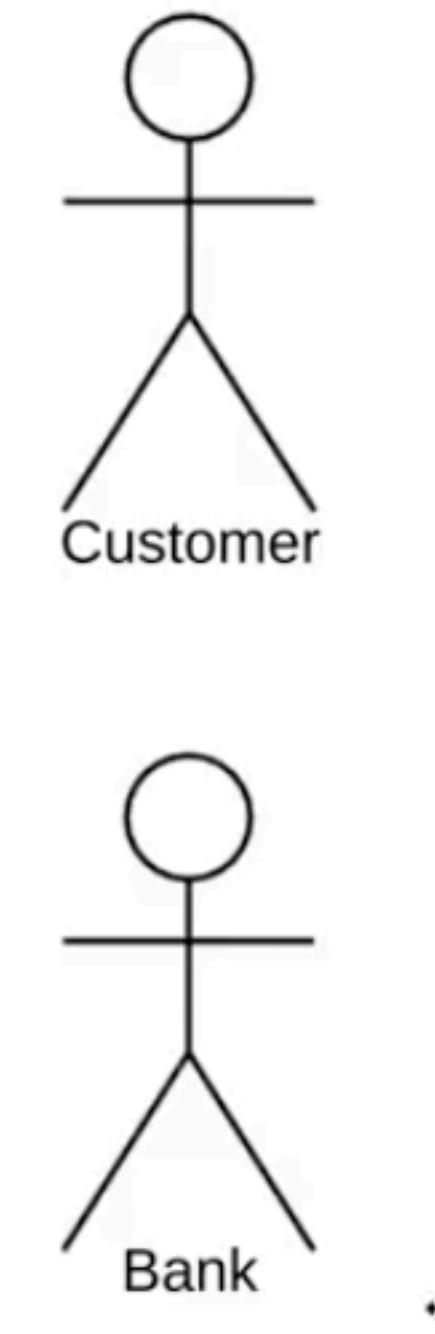
- Identificado pelo “boneco” desenhado junto ao sistema;
- Trata-se de algo ou alguém que usa o sistema a fim de atingir um objectivo;
- Poderá ser destacado como:
 - Pessoa;
 - organização;
 - Outro sistema;
 - Equipamento externo.
- De forma genérica será quem ou o quê que deverá utilizar o nosso sistema.



DESENVOLVIMENTO COLABORATIVO - UML

ACTOR (actor) - Elemento que influencia o sistema.

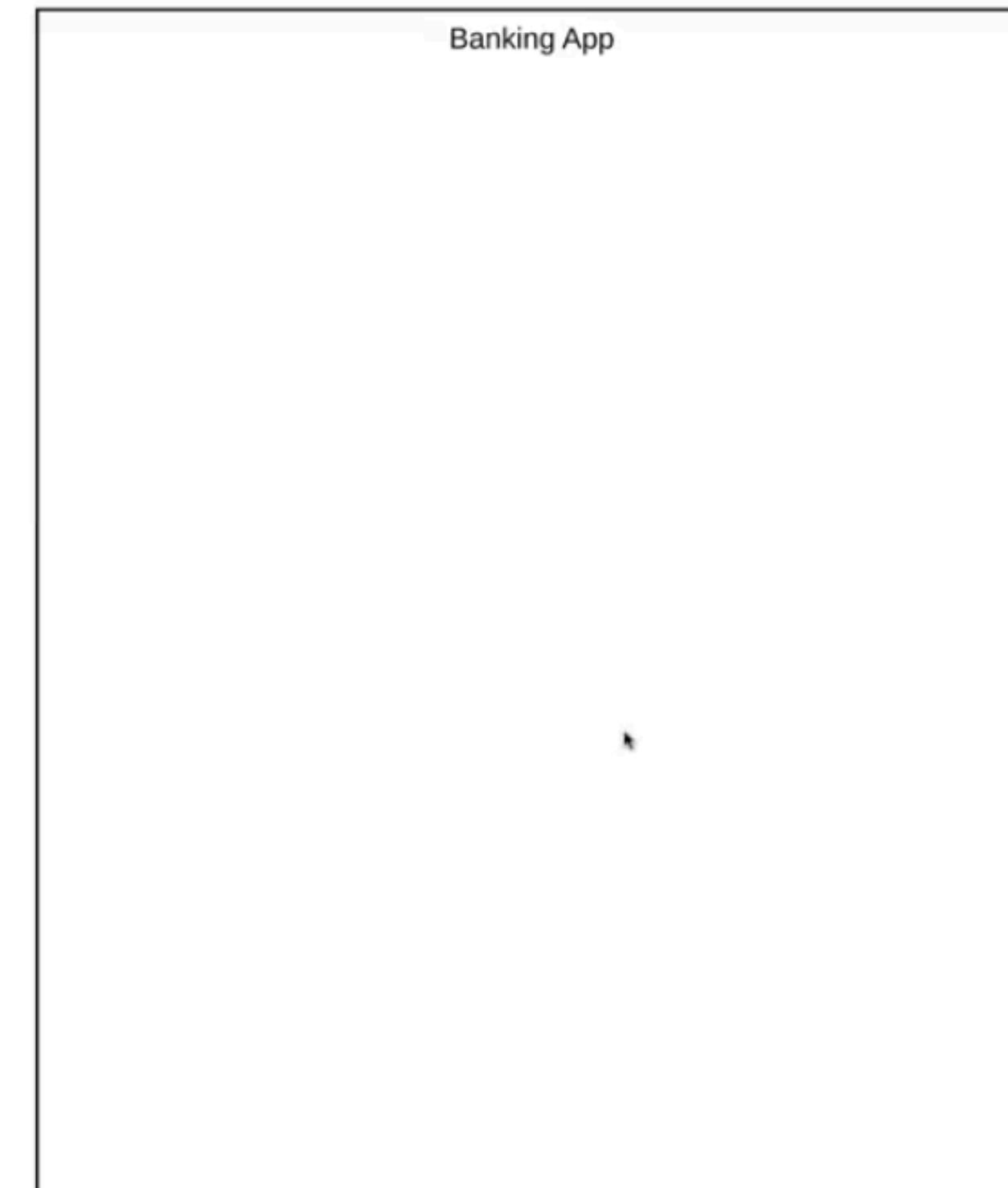
- Existem dois tipos de actores:
 - **Primário:** Alguém ou algo que utiliza o nosso sistema
Alguém ou algo que inicia o nosso sistema.
 - **Secundário:** Alguém ou algo que providencia informações do sistema ao actor principal.
Alguém ou algo que age mediante certa ação proveniente do actor principal.



DESENVOLVIMENTO COLABORATIVO - UML

ACTOR (actor) - Elemento que influencia o sistema.

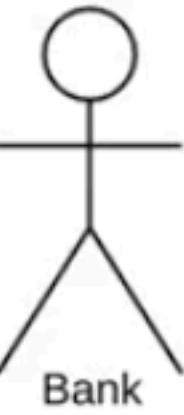
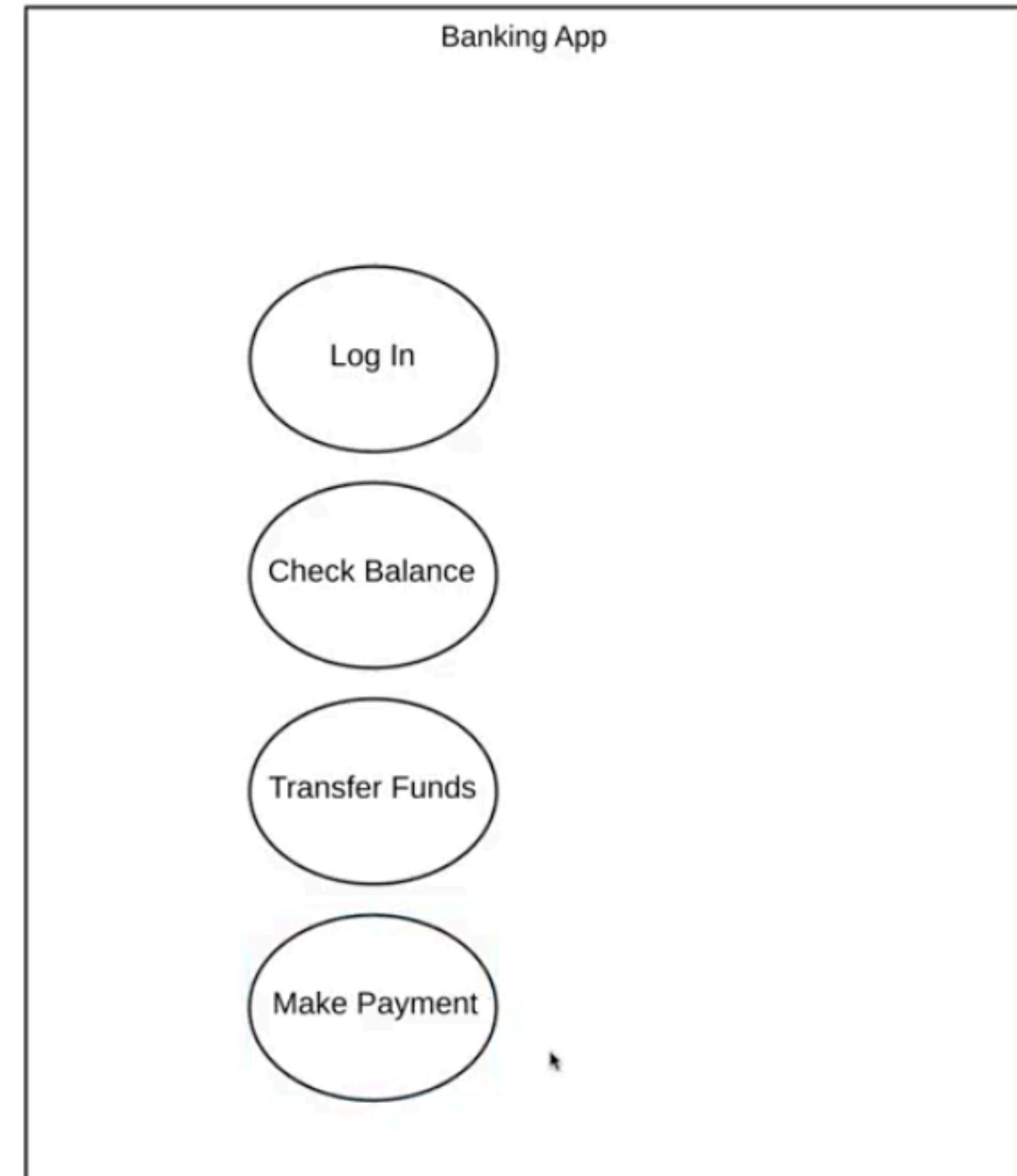
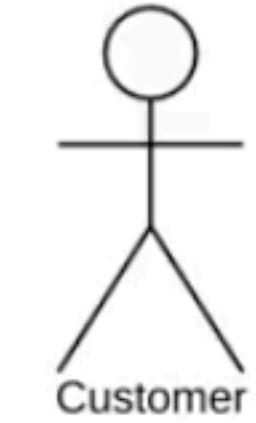
- Posicionamento de cada actor:
 - **Primário:** Esquerda do sistema
Inicializa o processo.
 - **Secundário:** Direita do sistema
Fornece a informação necessária.



DESENVOLVIMENTO COLABORATIVO - UML

Use CASE - Elemento que descreve as funcionalidades de cada sistema.

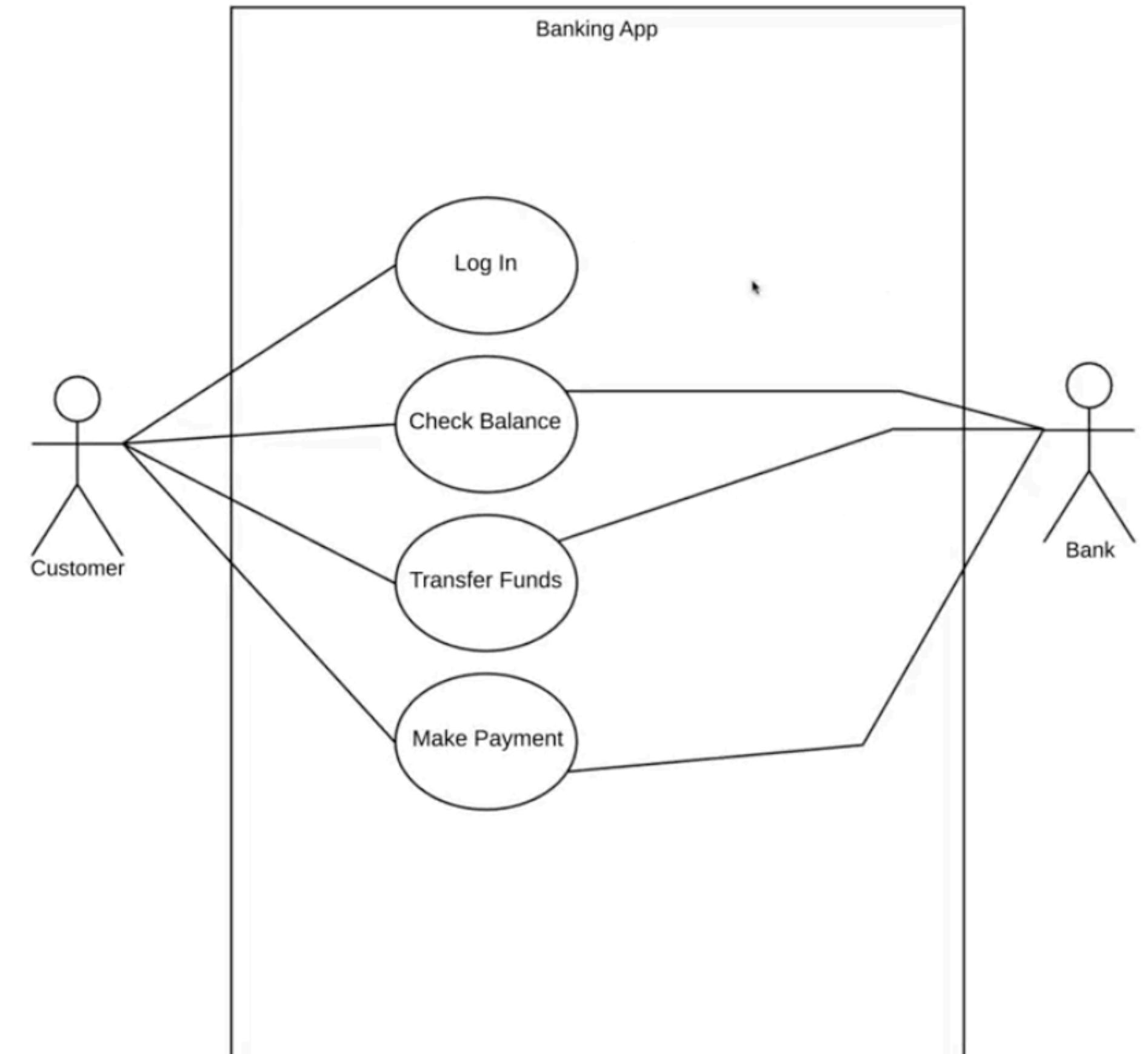
- Identificado por ser oval e cada forma representa uma funcionalidade



DESENVOLVIMENTO COLABORATIVO - UML

Relacionamentos - Relacionamentos entre os actores e as funcionalidades.

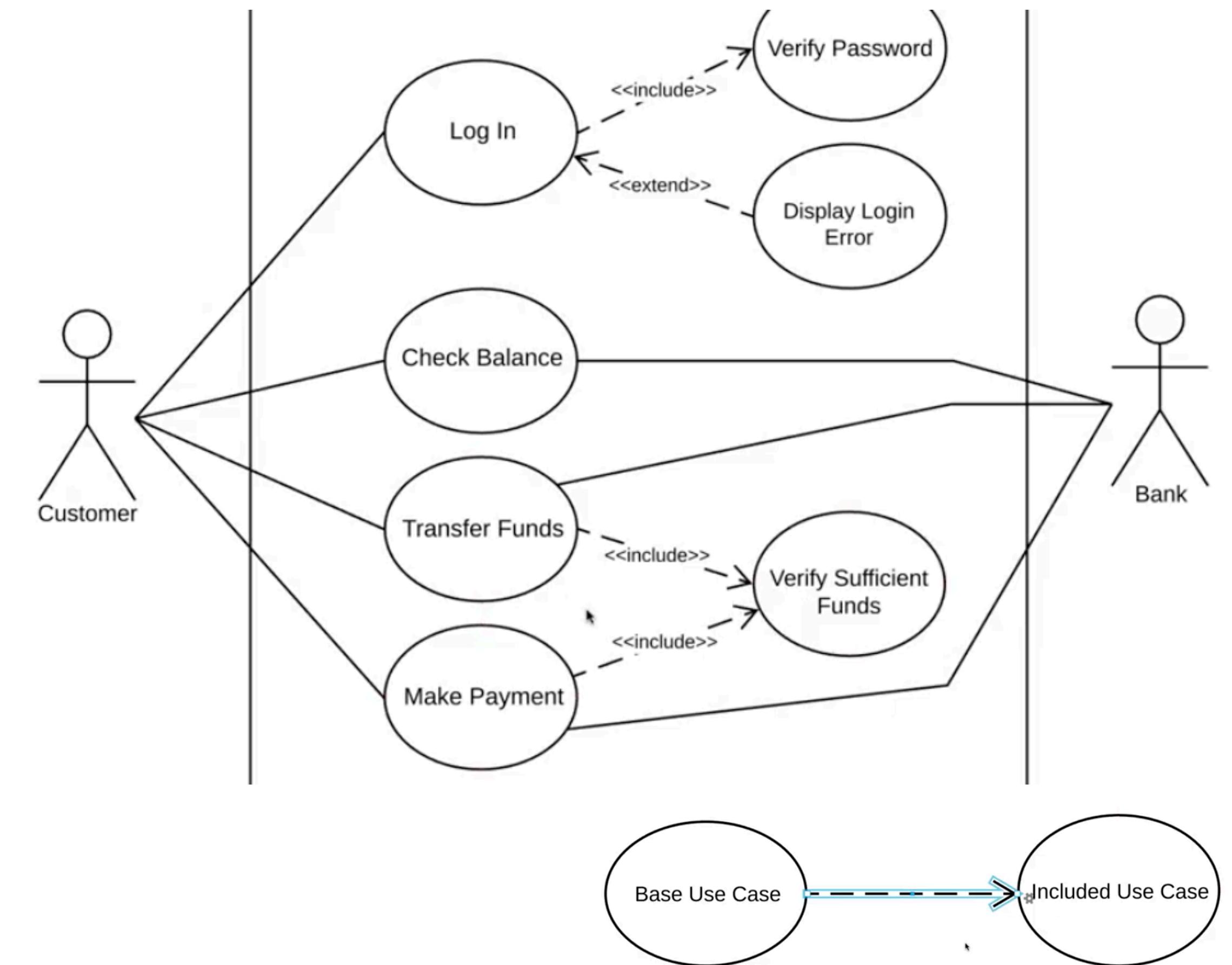
- **Associação** - relação de funcionalidade da task - Linha solida;
- **Include** - ?
- **Extend** - ?
- **Generalization** - ?



DESENVOLVIMENTO COLABORATIVO - UML

Relacionamentos - Relacionamentos entre os actores e as funcionalidades.

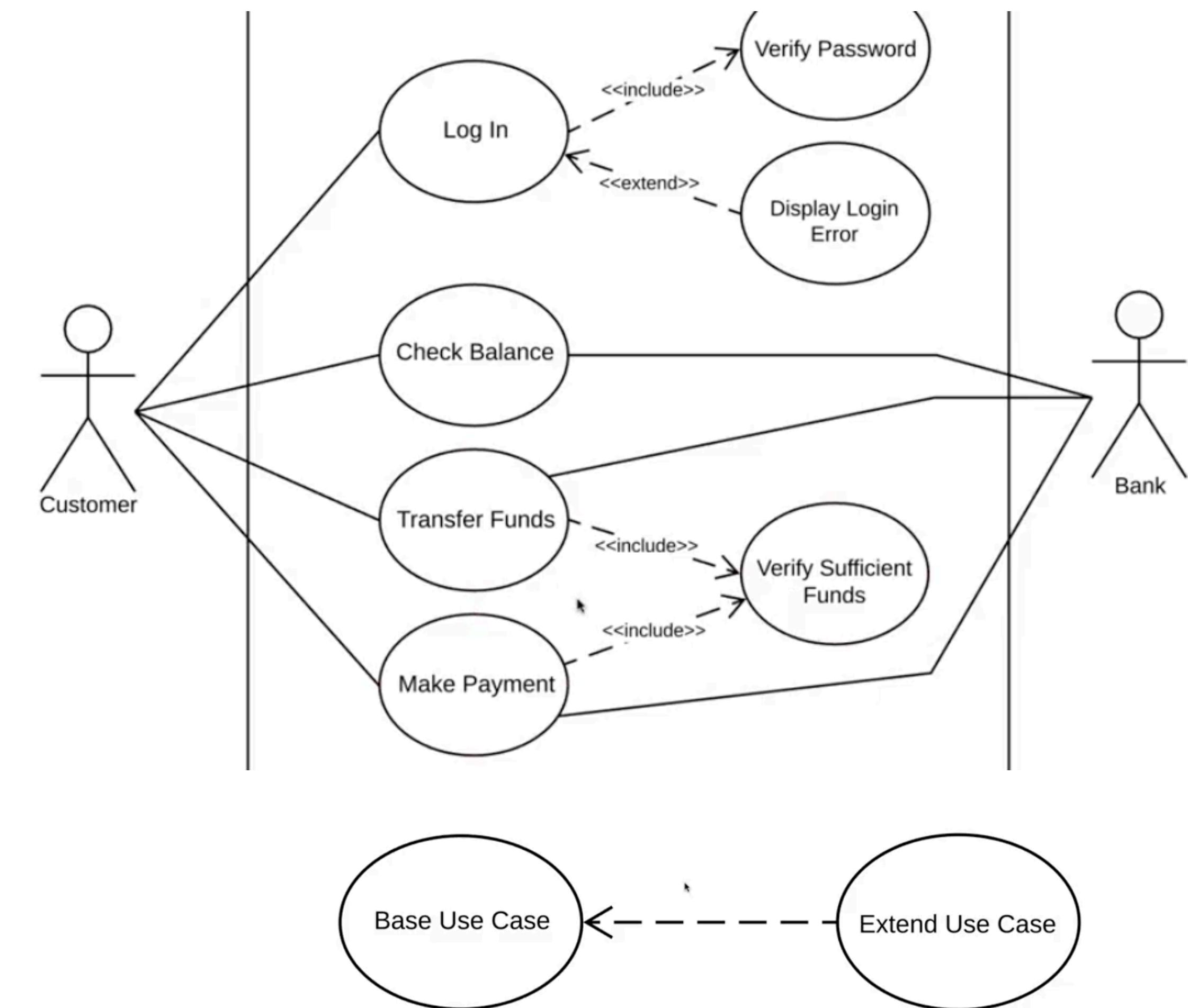
- Associação - relação de funcionalidade da task - Linha solida;
- **Include** - Apresenta dependencia de um use case included. Outra maneira de compreender este relacionamento passa por o use case só se torna completo quando o use case included esta terminado.
- Extend - ?
- Generalization - ?



DESENVOLVIMENTO COLABORATIVO - UML

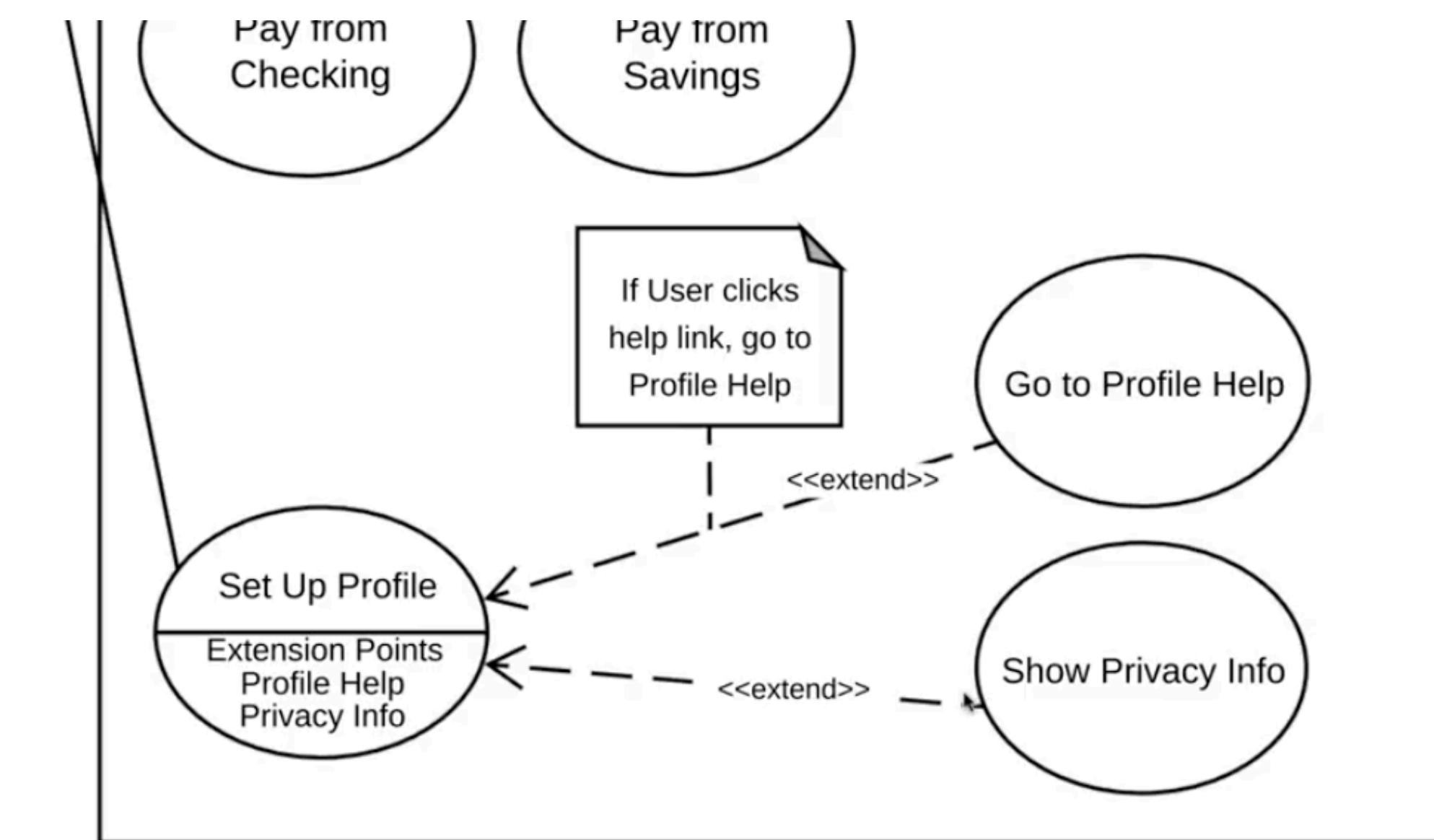
Relacionamentos - Relacionamentos entre os actores e as funcionalidades.

- **Associação** - relação de funcionalidade da task - Linha solida;
- **Include** - Apresenta dependencia de um use case included. Outra maneira de compreender este relacionamento passa por o use case só se torna completo quando o use case included esta terminado.
- **Extend** - Ocorre apenas algumas vezes. Para ocorrer tem que acertar em todos os requisitos.
- **Generalization** - ?

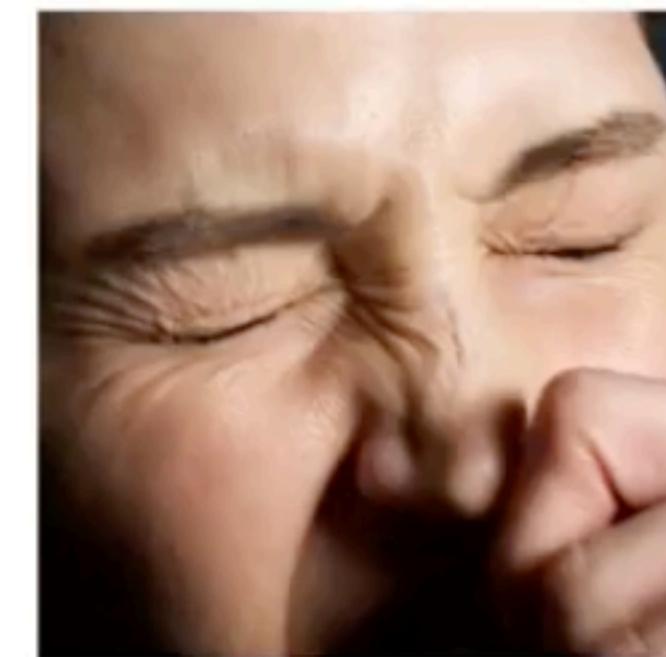
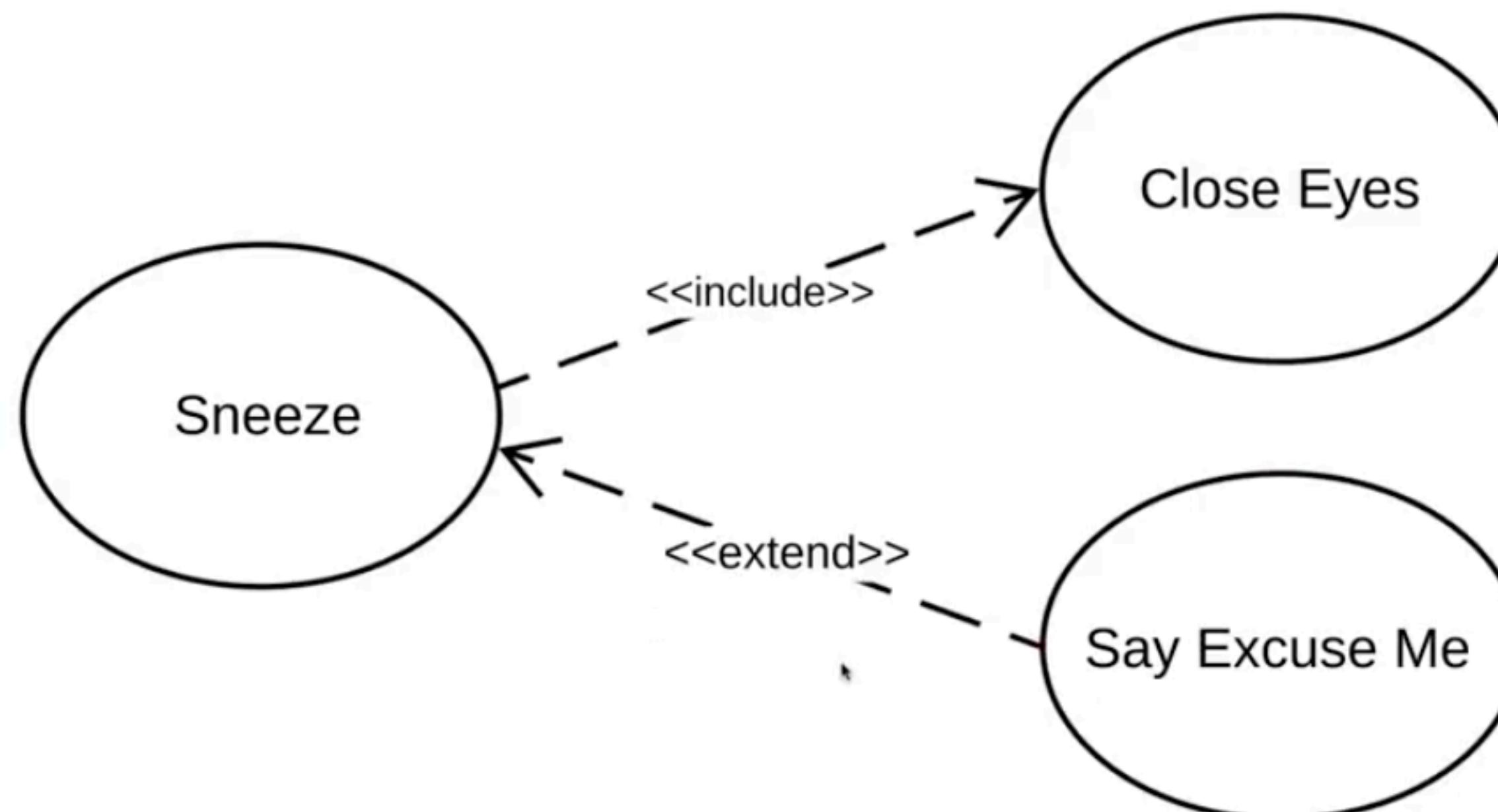
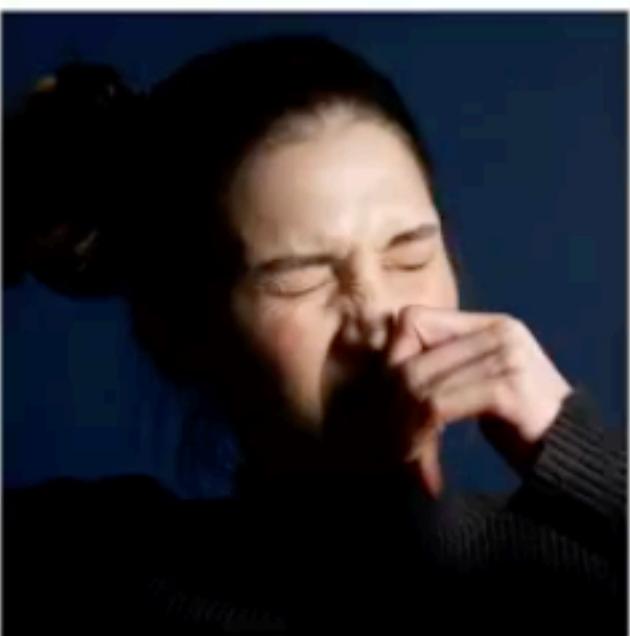


DESENVOLVIMENTO COLABORATIVO - UML

- **Extend - Extension Points** determinam:
 - Use cases que apresentam mais do que uma funcionalidade.
 - Este use case extensão point apresenta a possibilidade de criar um perfil bem como:
 - Apresenta ajuda;
 - Informações de privacidade.



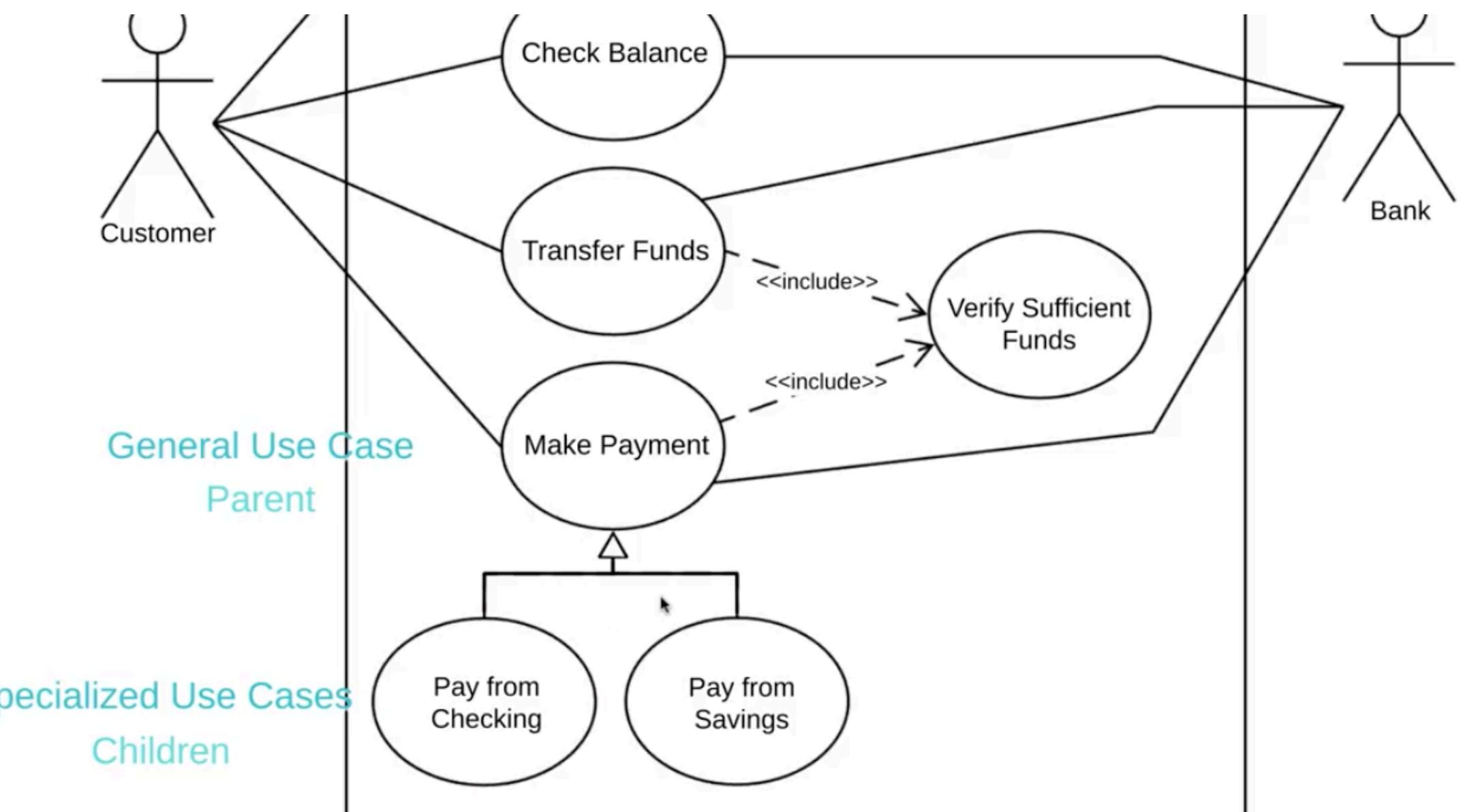
DESENVOLVIMENTO COLABORATIVO - UML



DESENVOLVIMENTO COLABORATIVO - UML

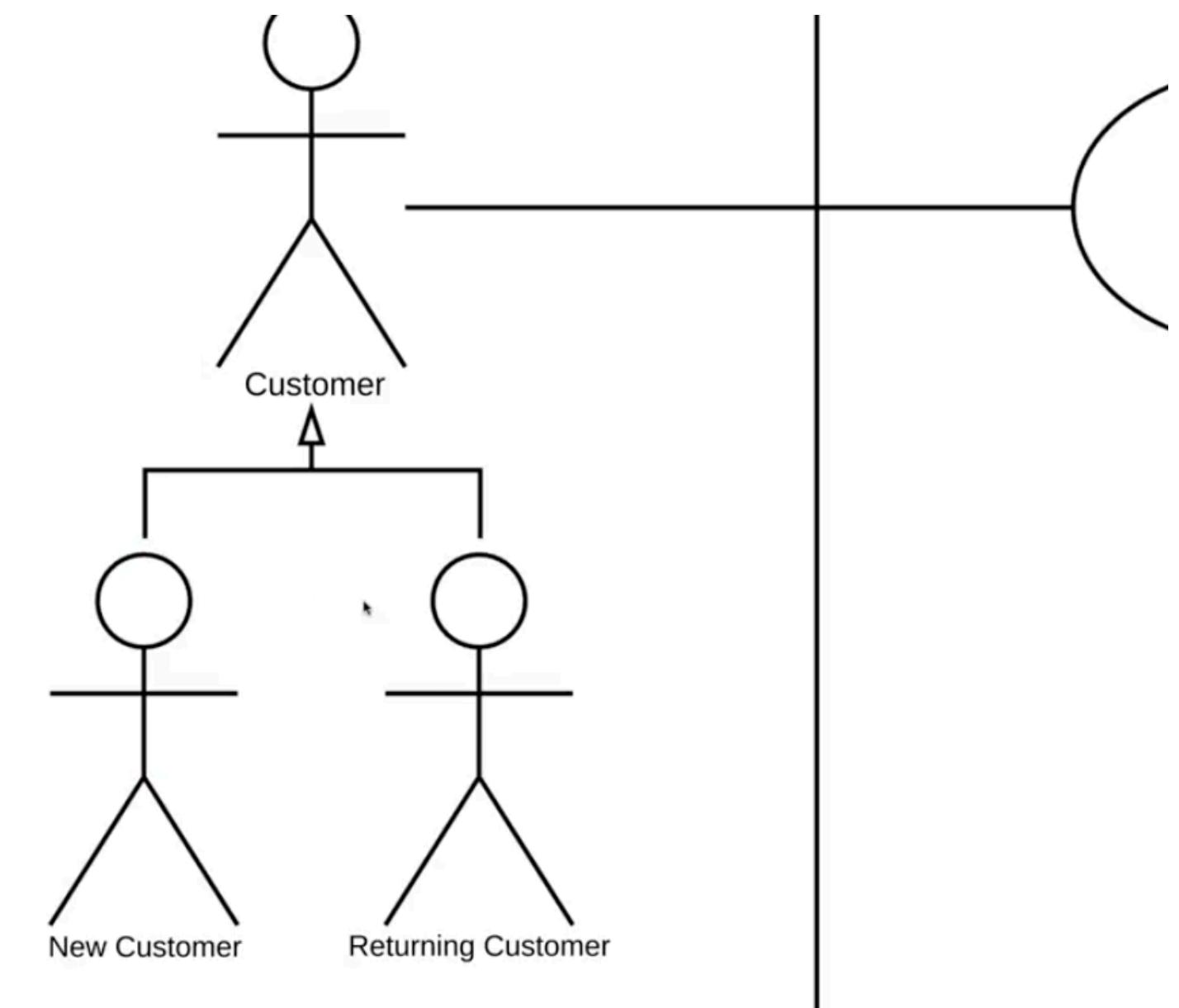
Relacionamentos - Relacionamentos entre os actores e as funcionalidades.

- **Associação** - relação de funcionalidade da task -
Linha solida;
- **Include** - Apresenta dependencia de um use case included. Outra maneira de compreender este relacionamento passa por o use case só se torna completo quando o use case included esta terminado.
- **Exclude** - Ocorre apenas algumas vezes. Para ocorrer tem que acertar em todos os requisitos.
- **Generalization** - Também denominados por use cases especializados. Basicamente tem o mesmo comportamento que os use cases normais contudo acrescentam mais valor ou seja mais opções.



DESENVOLVIMENTO COLABORATIVO - UML

- **Generalization** - Também denominados por use case especializados. Basicamente tem o mesmo comportamento que os use cases normais contudo acrescentam mais valor ou seja mais opções.
- Em certos cenários também podemos implementar esta opção ao nível dos actores



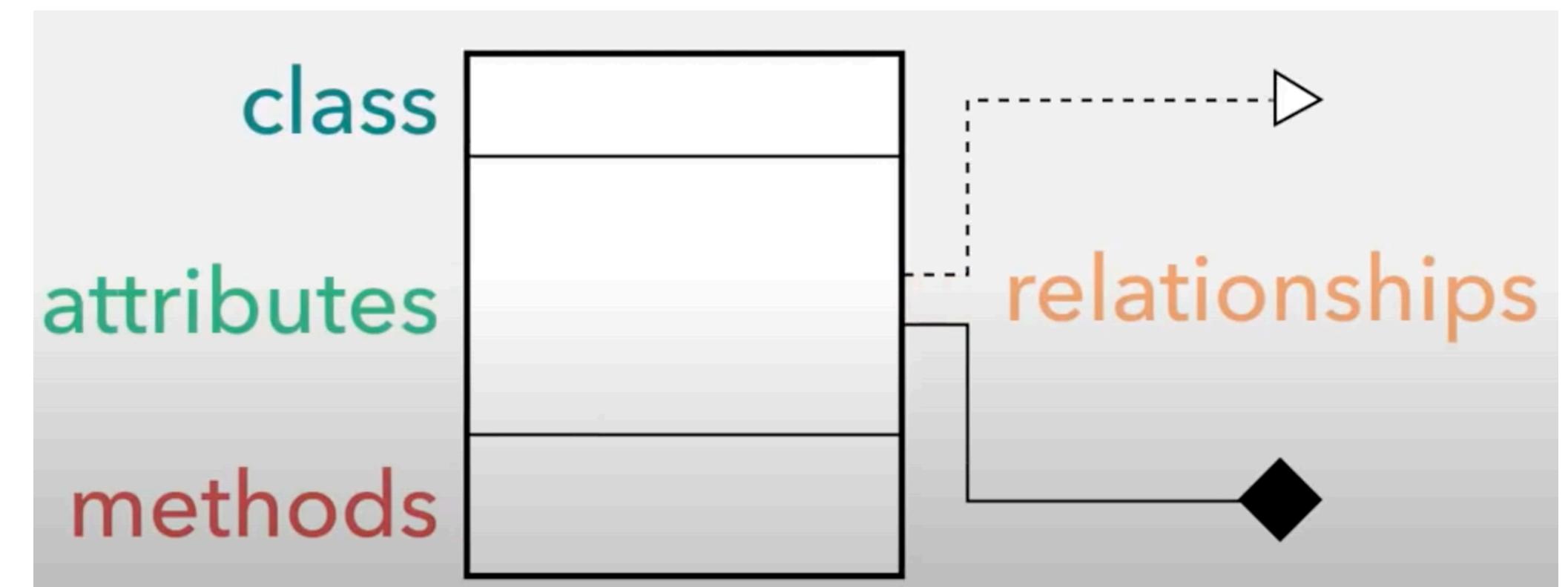
FUNDAMENTOS DE SQL

Diagramas de classes

DIAGRAMAS DE CLASSE

O esquema de diagrama de classes compreende as seguintes características:

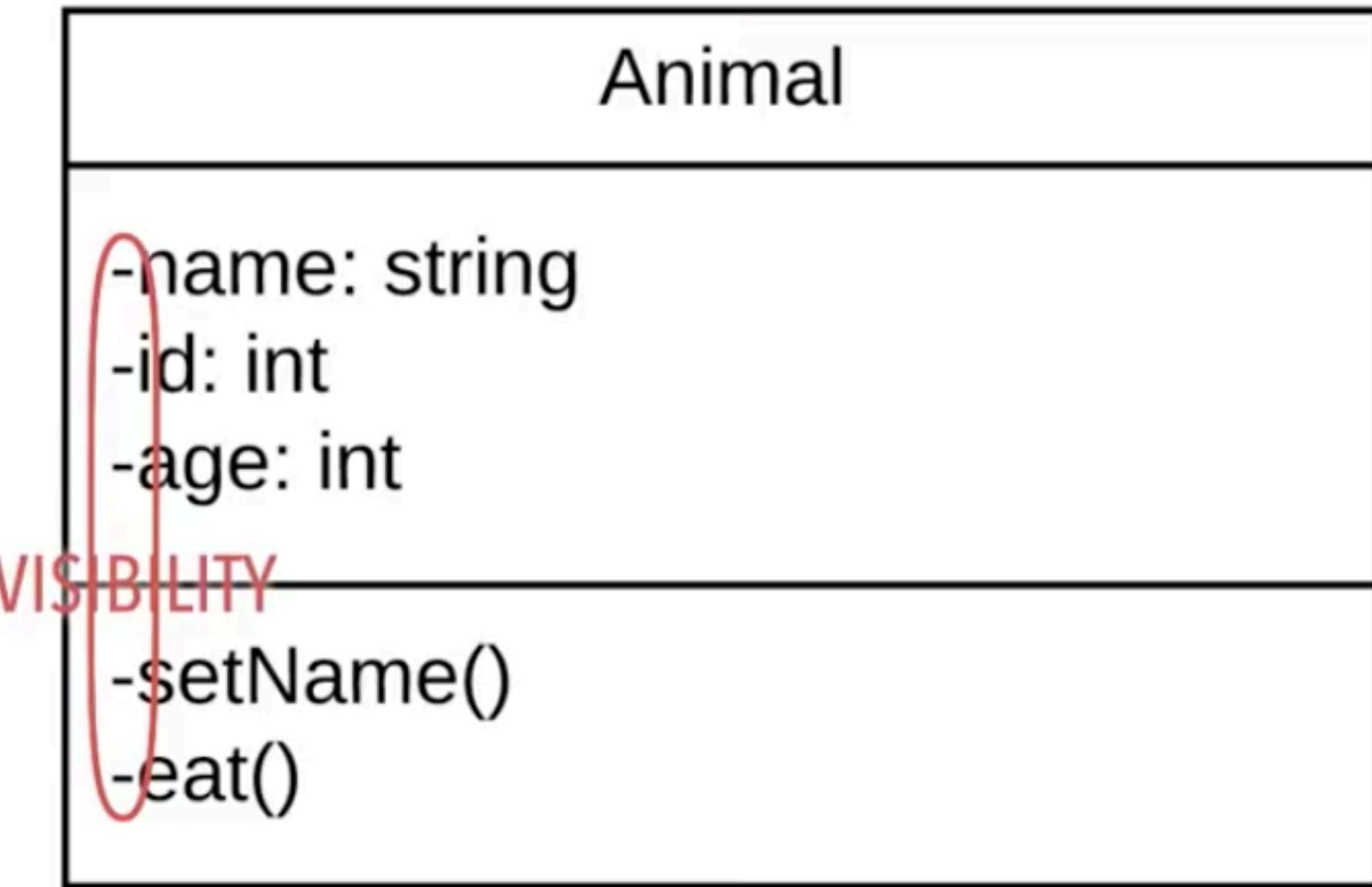
- **Classe** - Identifica/Determina algo
- **Atributos** - informação que compreende e descreve a instancia da classe (variáveis, campos ou propriedades)
.Constituído por um hífen antes da propriedade e dois pontos depois da mesma e de seguida o tipo de dados. (Exemplo: -nome:string)
- **Métodos** - Operações ou funções. Permite especificar qualquer tipo de comportamento, característica de uma classe (Exemplo: -eat())
- **Relações** - Relacionamento entre classes



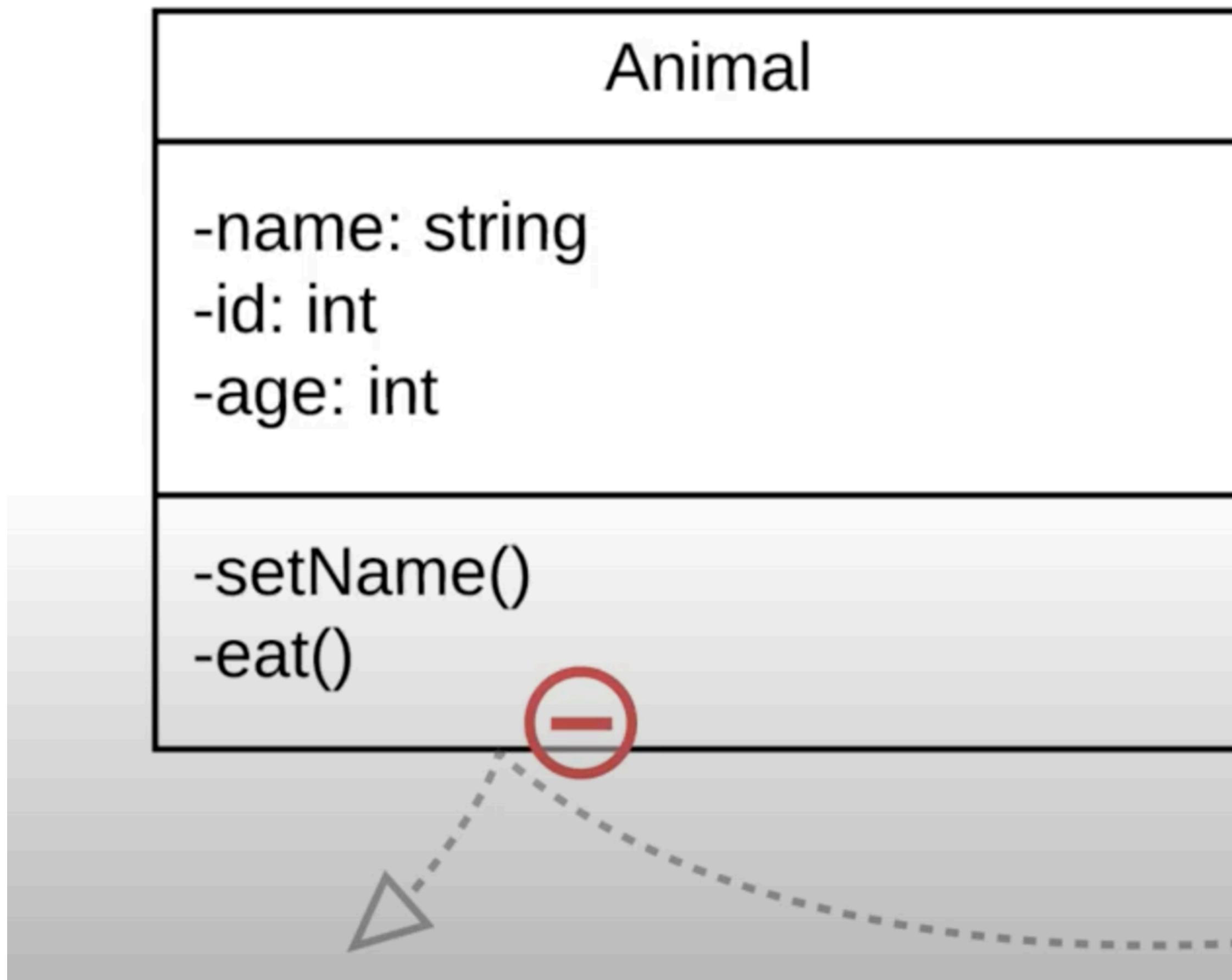
DIAGRAMAS DE CLASSE - VISIBILIDADE

A visibilidade trata-se de uma comportamento inerente às classes sendo que permite ou não que as mesmas possam interagir umas com as outras ou seja:

- **privado** -> as classes não conseguem interagir entre si
- + **publico** -> as classes conseguem interagir entre si
- # **protected** -> as classes conseguem interagir entre si, mas dentro da mesma família de classe e subclasse
- ~ **package/default** -> as classes conseguem interagir entre si desde que estejam dentro da mesma classe.

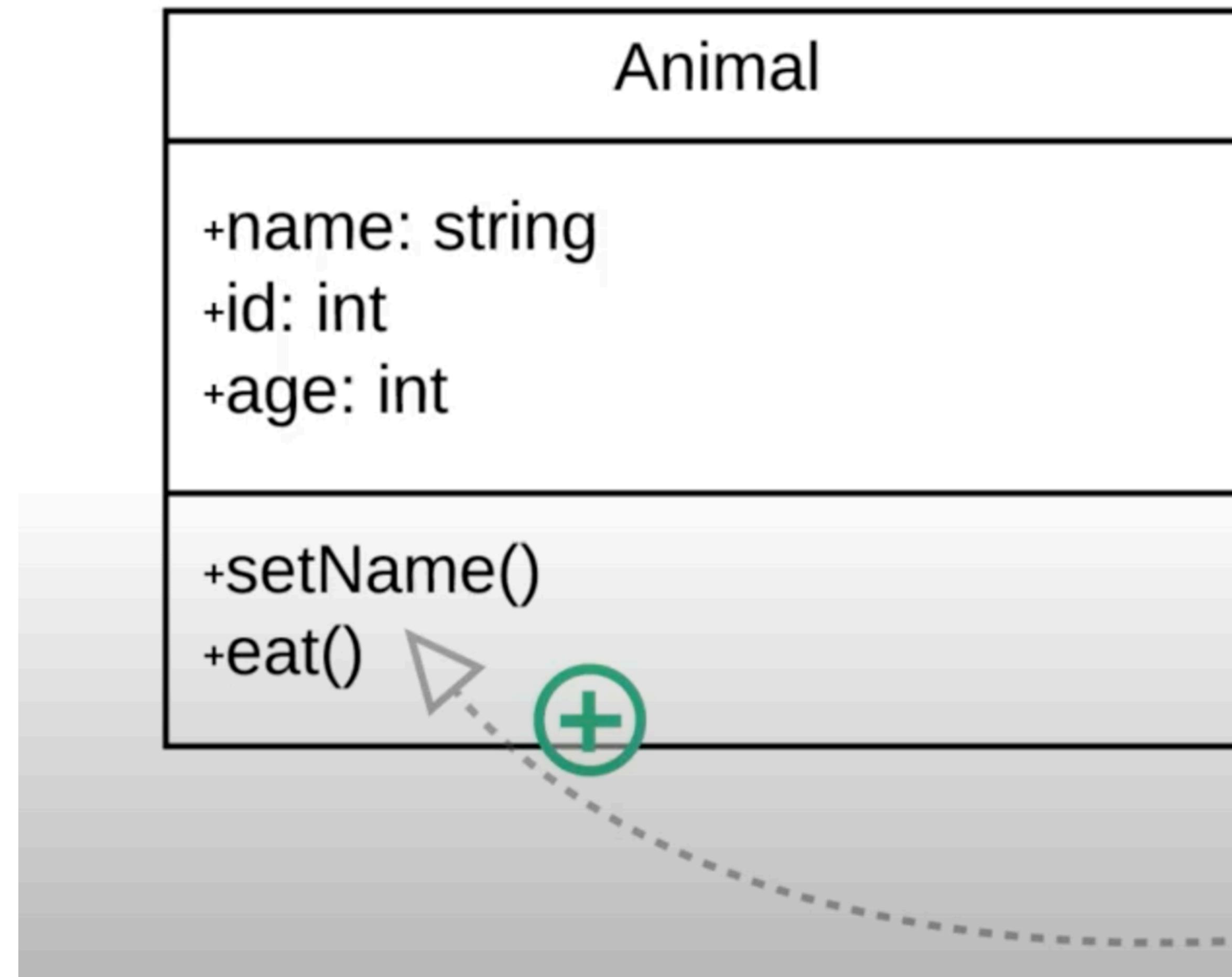


DIAGRAMAS DE CLASSE - VISIBILIDADE (- PRIVADO)



Visibility
- private

DIAGRAMAS DE CLASSE - VISIBILIDADE (+ PUBLICO)



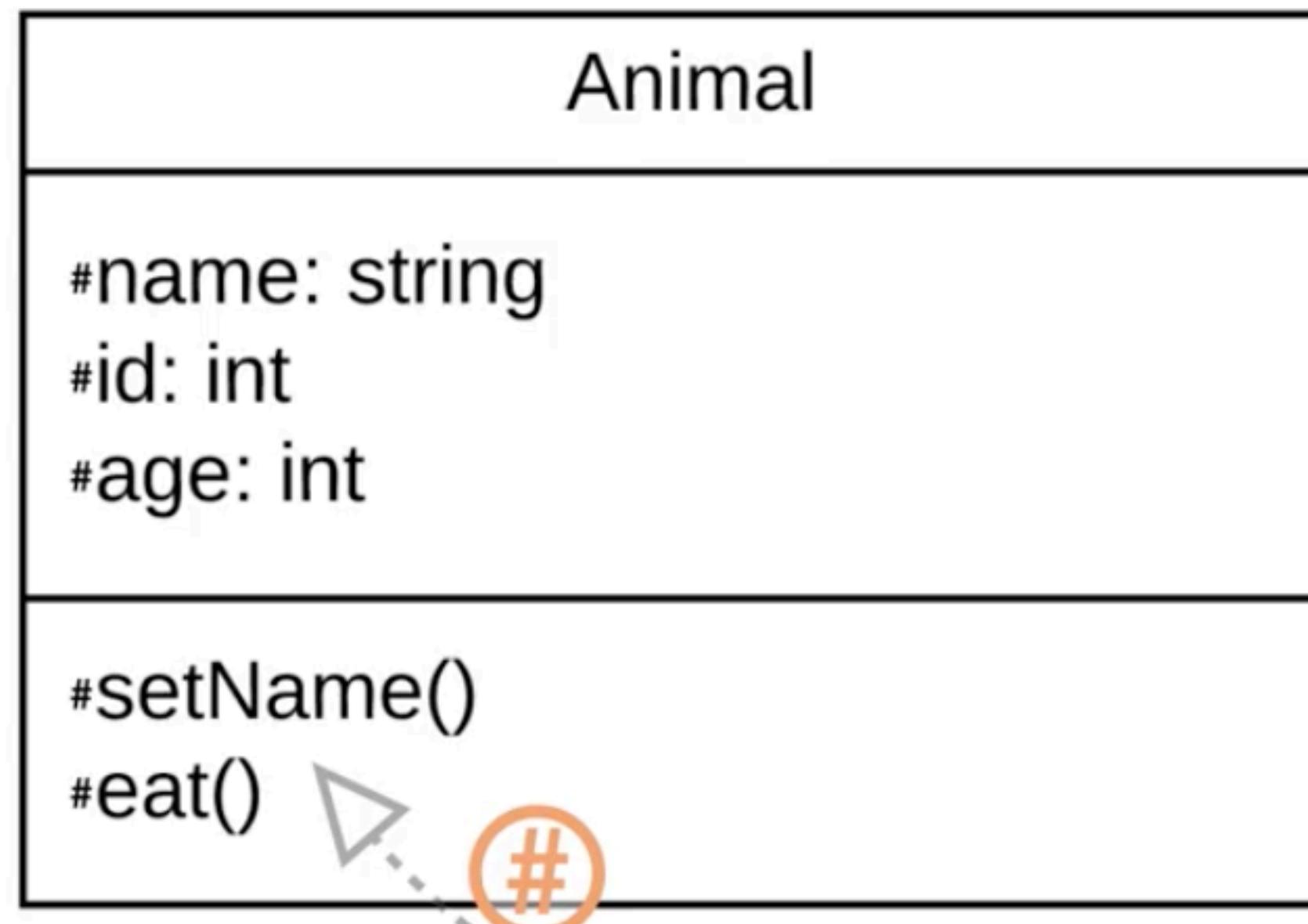
Visibility

- private
+ public



DIAGRAMAS DE CLASSE - VISIBILIDADE (# PROTECTED)

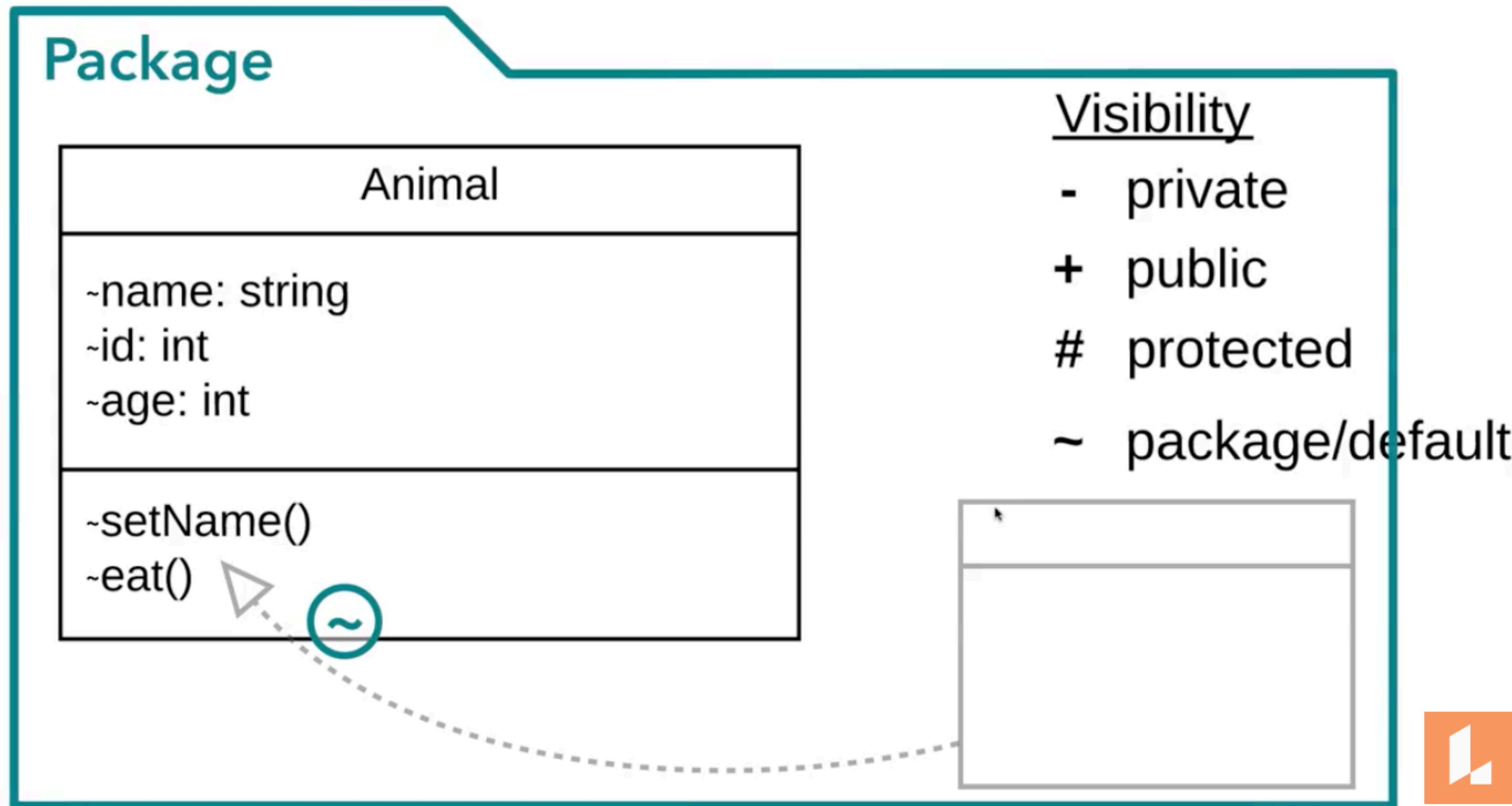
Zoo System



Visibility

- private
- + public
- # protected

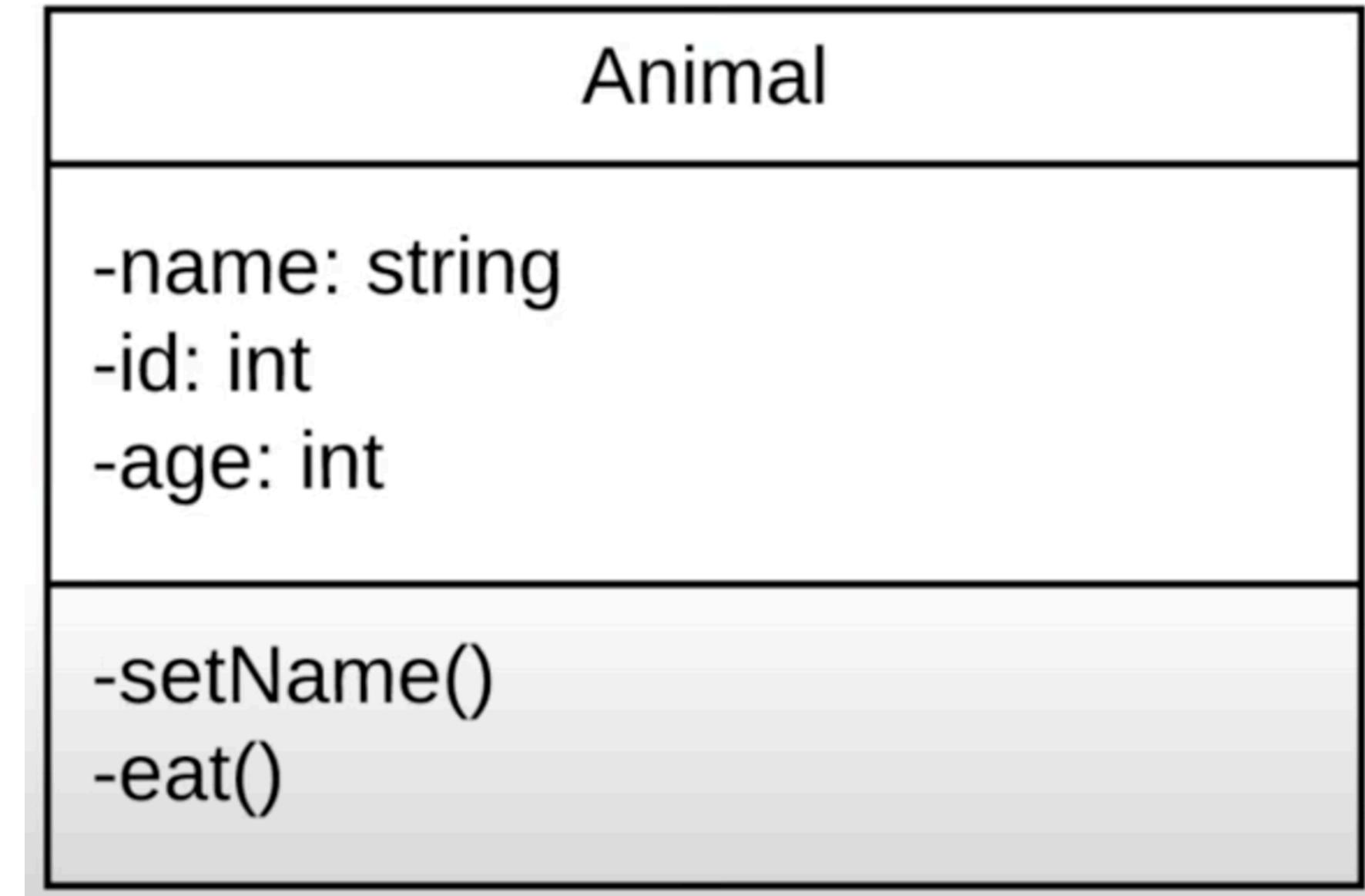
DIAGRAMAS DE CLASSE - VISIBILIDADE (~ PACKAGE)



DIAGRAMAS DE CLASSE - EXEMPLO JARDIM ZOOLÓGICO

Exemplo: No Jardim zoológico temos que identificar:

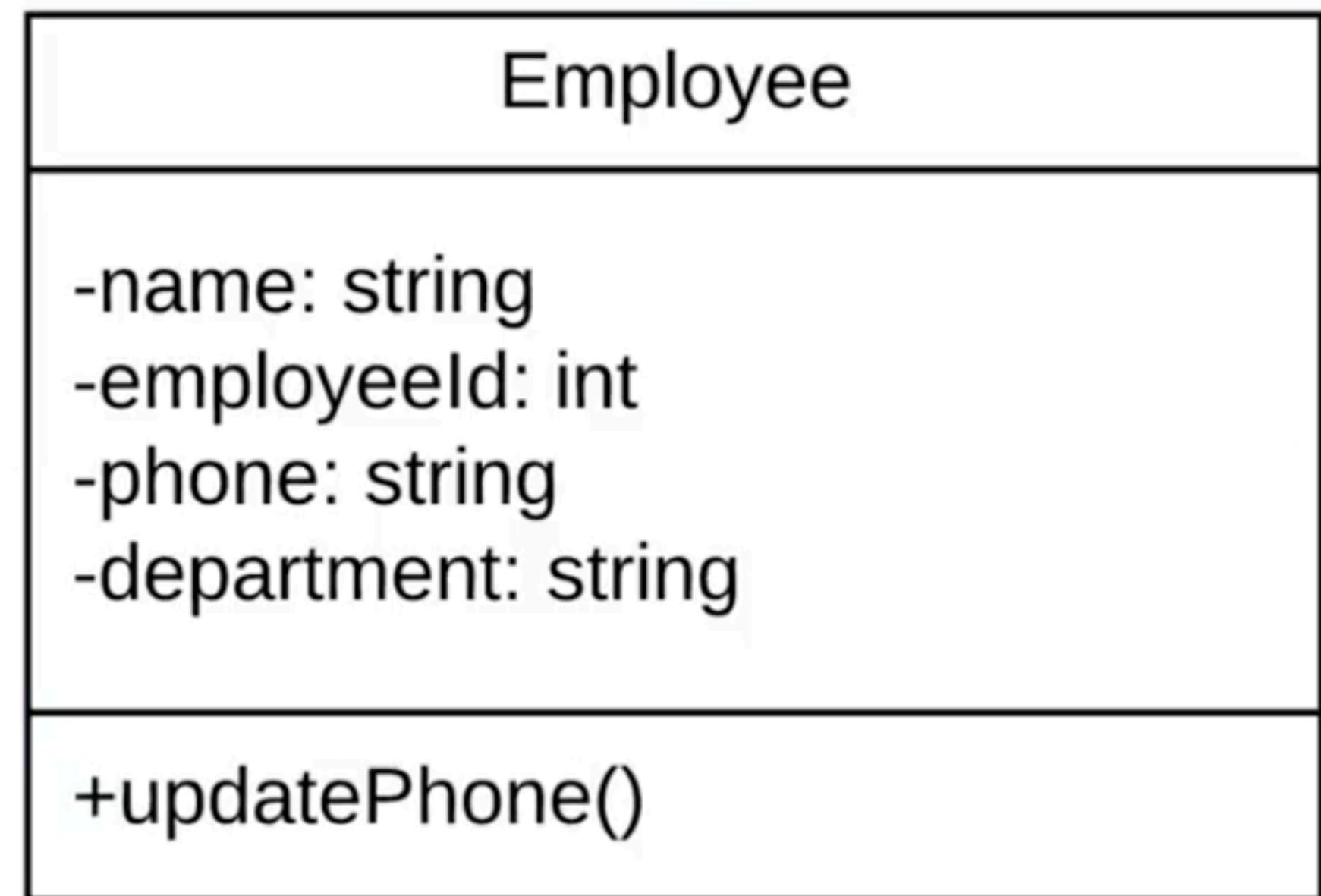
- ➊ Classe - O que existe o Jardim Zoológico?
- ➋ Atributos - Informações de um determinado animal.
- ➌ Método - Quais os comportamentos do animal?



DIAGRAMAS DE CLASSE - EXEMPLO EMPRESA

Exemplo: Numa empresa todos os trabalhadores são identificados:

- **Classe** - Tipo de pessoa inerente à empresa?
- **Atributos** - Informações da pessoa.
- **Método** - Quais as informações/comportamentos que podem ser alterados inerente à pessoa?



DIAGRAMAS DE CLASSE - RELACIONAMENTOS

Relacionamentos:

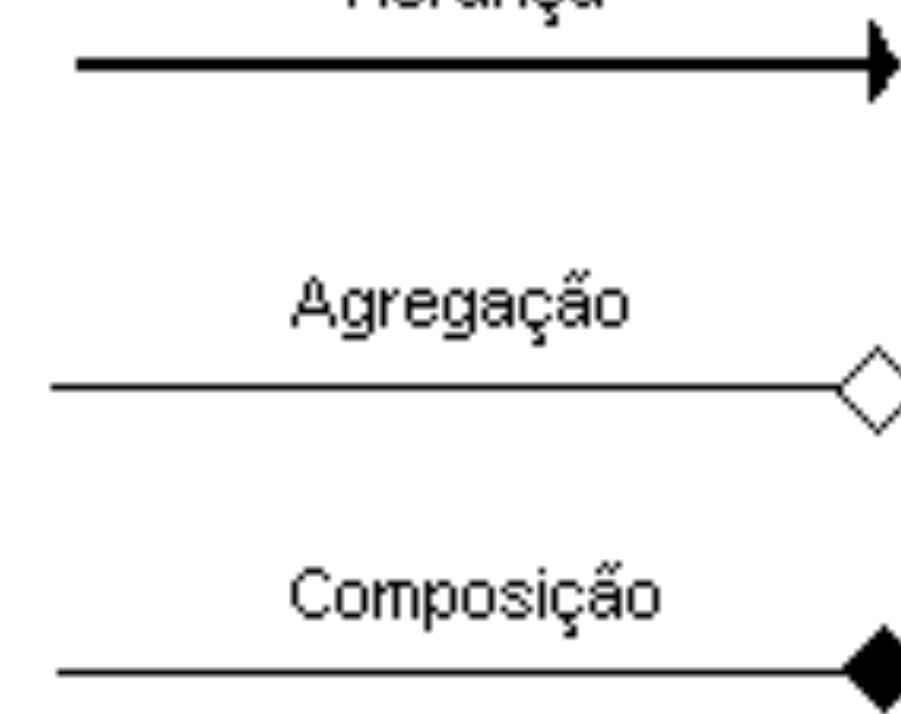
- Associação;
- Herança;
- Agregação;
- Composição

Associação

Herança

Agregação

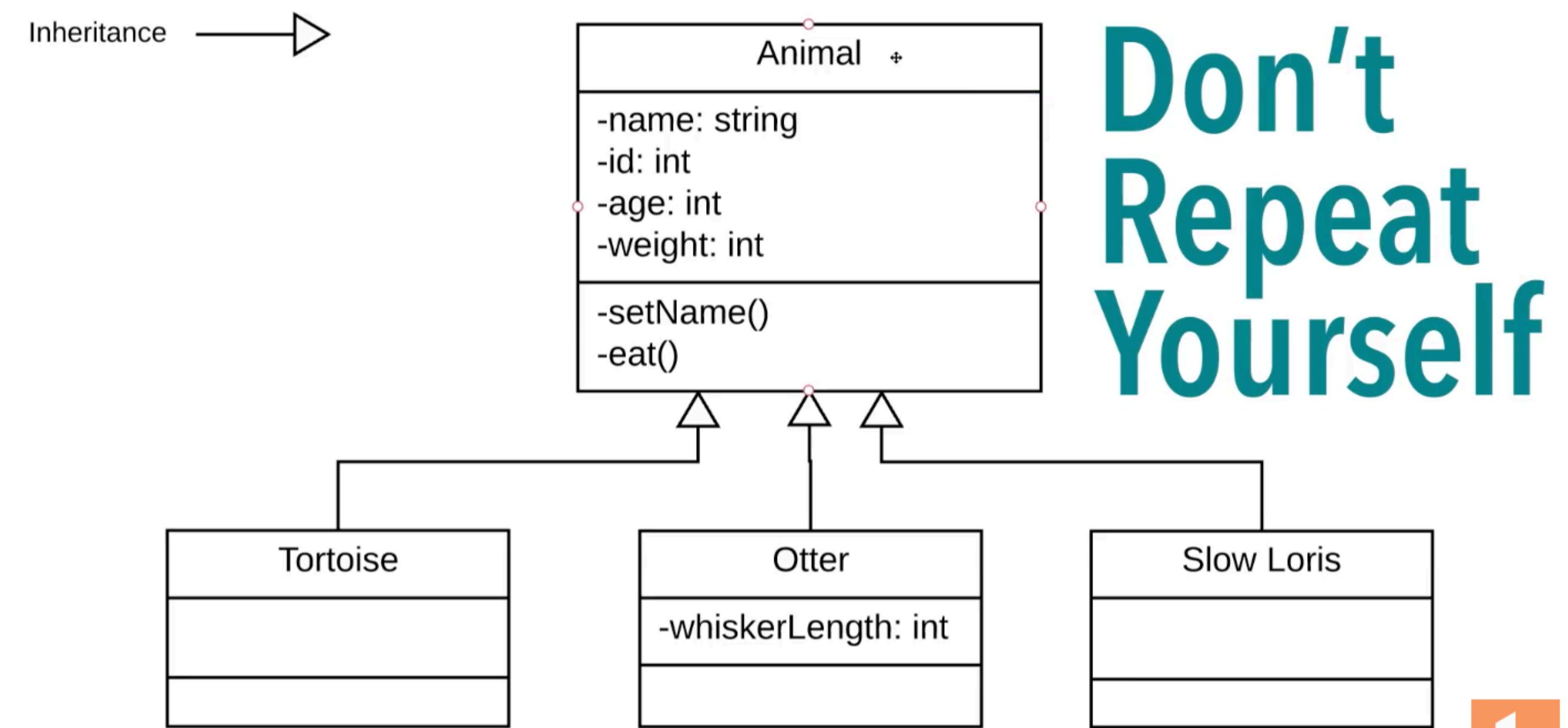
Composição



DIAGRAMAS DE CLASSE - RELACIONAMENTO DE HERANÇA

Relação de herança:

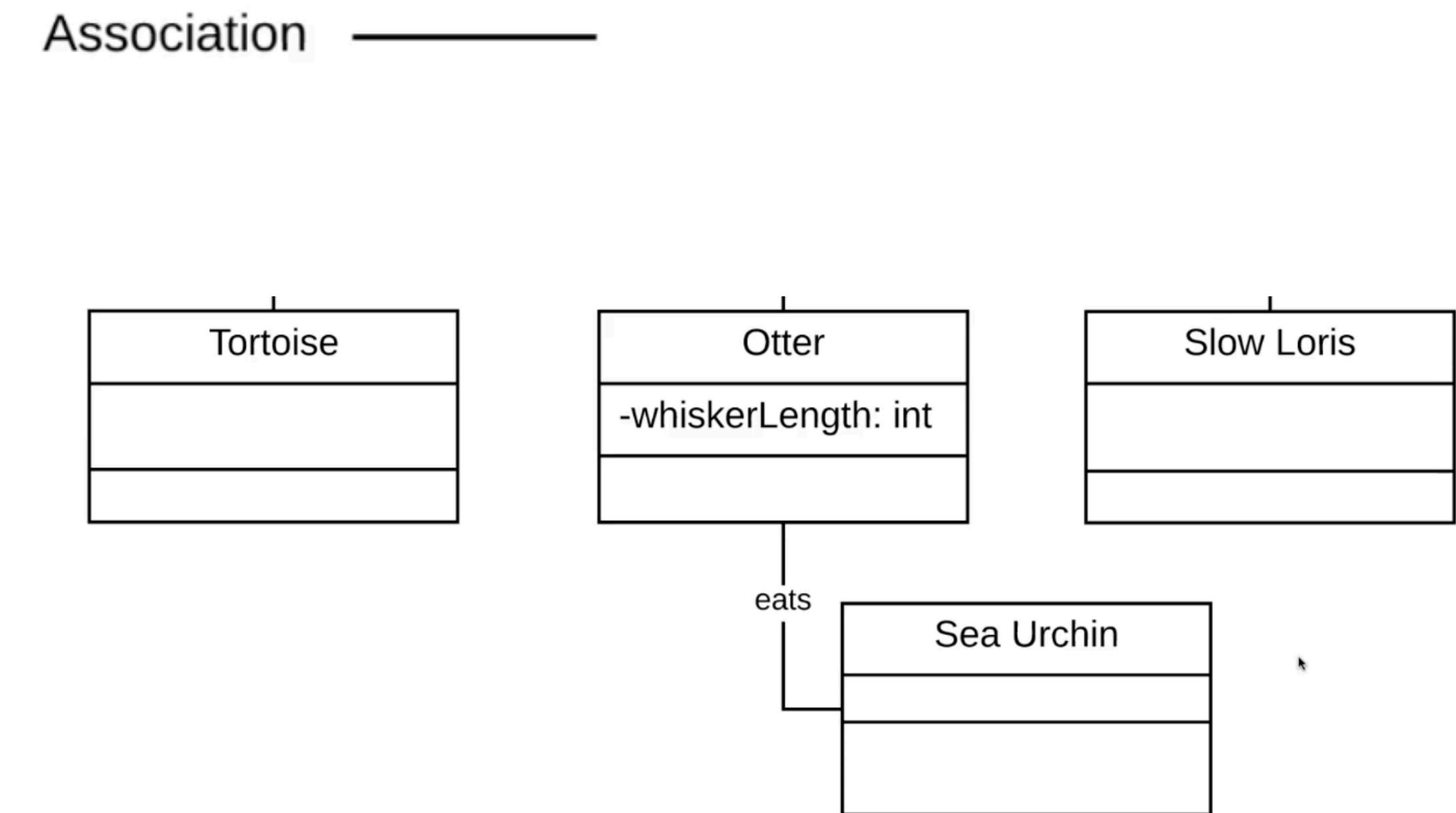
- Existe a **SUPERCLASSE** que compreendo o global do que estamos a identificar;
- Existe a **SUBCLASSE** que apresenta uma herança da classe-mãe/ Superclasse;
- Estas subclasses comprehendem características específicas (neste caso específico de cada animal).



DIAGRAMAS DE CLASSE - RELACIONAMENTO DE ASSOCIAÇÃO

Relação de associação:

- Existe uma relação entre uma subclasse e uma sub-subclasse onde aborda uma dependência;
- Neste caso específico a Lontra (OTTER) come os ouriços-do-mar (Sea Urchin)

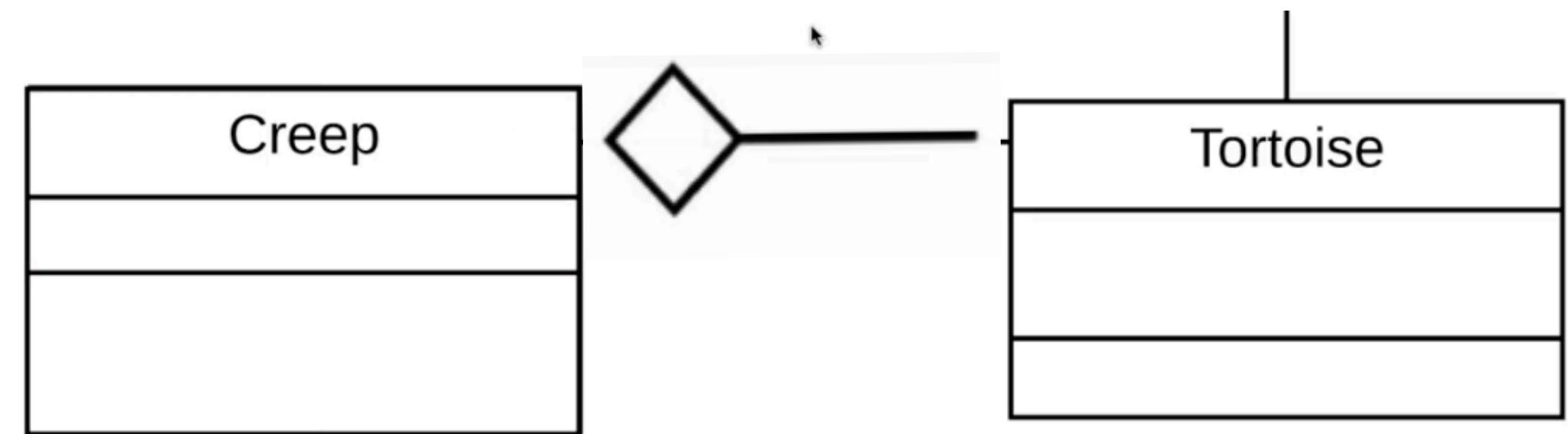


DIAGRAMAS DE CLASSE - RELACIONAMENTO DE AGREGAÇÃO

Relação de agregação:

- Existe uma relação entre uma subclasse e uma sub-subclasse onde aborda uma dependência;
- Neste caso específico um grupo de tartarugas denomina-se por “bando” (CREEP) em que poderá fazer parte várias tartarugas contudo uma tartaruga pode abandonar o grupo e ficar solitária, logo não existe dependência entre ambas.

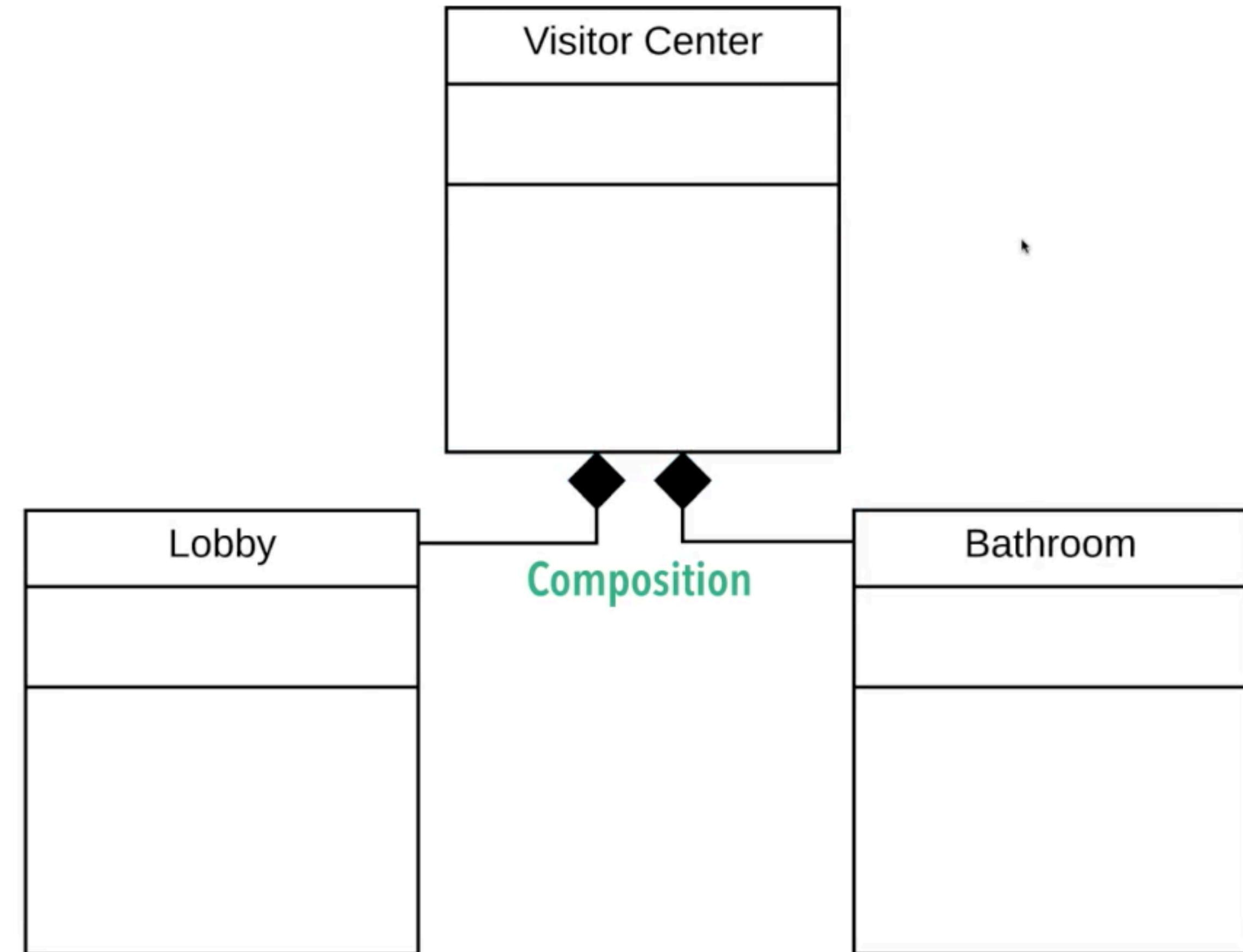
Aggregation



DIAGRAMAS DE CLASSE - RELACIONAMENTO DE COMPOSIÇÃO

Relação de Composição:

- Existe uma relação entre uma subclasse e uma sub-subclasse onde aborda uma dependência;
- Neste caso específico de o centro de visita (VISITOR CENTER) for demolido, a entrada (LOBBY) e casa de banho (BATHROOM) terá que ser automaticamente demolida também.



DIAGRAMAS DE CLASSE - MULTIPLICAÇÃO

Multiplicity

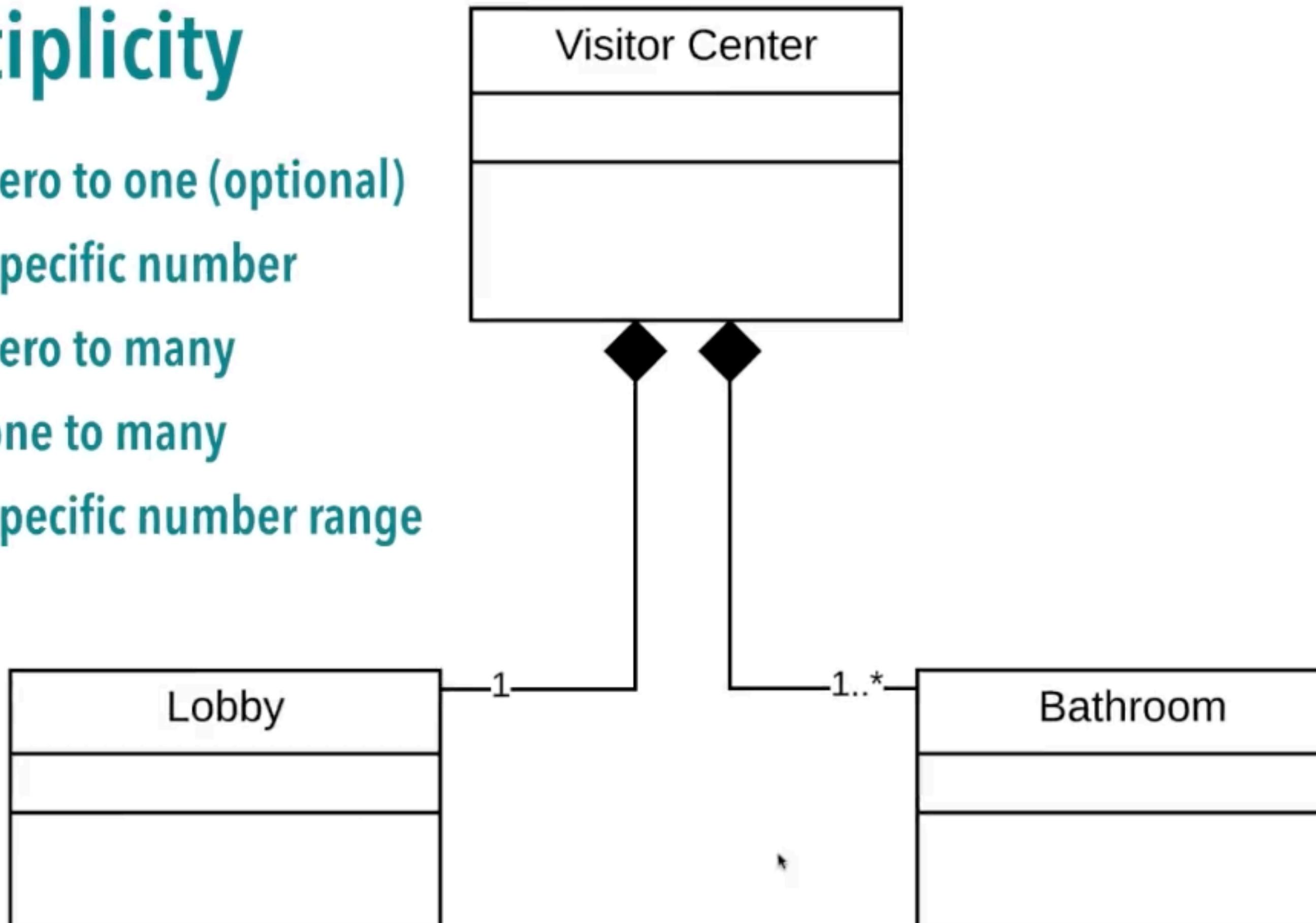
0..1 zero to one (optional)

n specific number

0..* zero to many

1..* one to many

m..n specific number range



FUNDAMENTOS DE SQL

Testes Aplicacionais

INTRODUÇÃO AOS TESTES APLICACIONAIS

O QUE É UM TESTE?

O termo teste vem do Latim, “testum” que significa, frasco de barro usado para ensaios com metais a fim de determinar a sua presença ou medir a massa de seus vários elementos.

Bem como este termo também surge na Lingua Inglesa quando mencionamos “put to the test”.

Ou seja, colocar um software em testes significar verificar a presença de defeitos.



INTRODUÇÃO AOS TESTES APLICACIONAIS

OBJECTIVOS DE UM TESTE?

A execução de um teste tem por objectivo verificar:

- Aspectos estruturais;
- Lógicos do software;
- Aspectos sistémicos.



Descobrir defeitos:

- A um custo adequado;
- Conceitos;
- Estratégias;
- Técnicas e Métricas.

INTRODUÇÃO AOS TESTES APLICACIONAIS - TÉCNICAS

As técnicas de verificação e validação podem ser divididas em:

- Estáticas;
- Dinâmicas.



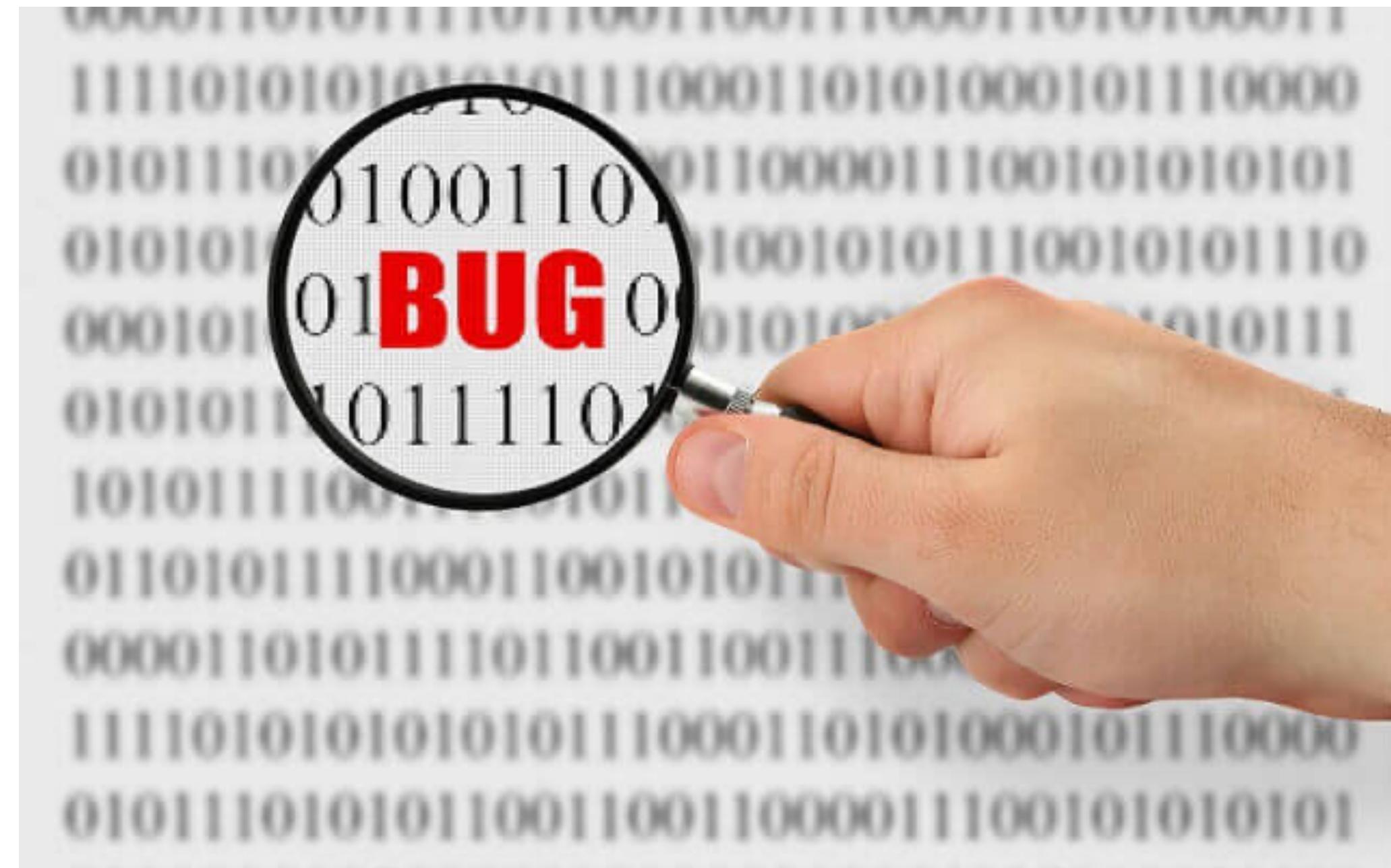
INTRODUÇÃO AOS TESTES APLICACIONAIS

QUE TIPOS DE TESTE DE SOFTWARE EXISTEM?

Os testes de software são uma maneira de garantir a qualidade de um software. Através disso existem diversas maneiras de se testar um sistema, que podem variar de acordo com a sua natureza ou seu objetivo.

Podemos classificar em:

1. **Testes estruturais ou caixa-branca (*white-box testing*):** teste de unidade e teste de integração.
2. **Testes funcionais ou caixa-preta (*black-box testing*):** teste de sistema e teste de aceitação.
3. **Testes não-funcionais:** teste de carga e teste de segurança.
4. **Testes de regressão:** teste de manutenção e teste de confirmação.



INTRODUÇÃO AOS TESTES APLICACIONAIS - TIPOS DE TESTE

Tipos de Teste

Teste Funcional

Teste Estrutural

Teste Não-Funcional

Teste de Regressão

- Teste de Sistema
- Teste de Aceitação

- Teste de Unidade
- Teste de Integração

- Teste de Usabilidade
- Teste de Carga
- Teste de Segurança

- Teste de Manutenção
- Teste de Confirmação

Técnicas de Teste

INTRODUÇÃO AOS TESTES APLICACIONAIS - TESTES ESTRUTURAIS

O QUE SÃO TESTES ESTRUTURAIS?

Testes estruturais ou testes de caixa-branca são:

- Realizados diretamente no código e geralmente são feitos pelo desenvolvedor do sistema;
- Exemplo deste tipo de teste são os:
 - **Testes de unidade (*unit testing*)** - Cada função do sistema é testada a fim de garantir que elas funcionam independentemente da interação com as outras partes do sistema;
 - **Teste de integração** - Estes testes tem o objetivo de verificar se cada parte do sistema, ao ser integrada com outras, funcionam corretamente.



INTRODUÇÃO AOS TESTES APLICACIONAIS - TESTES FUNCIONAIS

O QUE SÃO TESTES FUNCIONAIS?

- Os testes caixa-preta ou testes funcionais são executados pelos elementos que testam o sistema, nomeadamente clientes ou utilizadores. Sendo que estes não têm qualquer contato com o código-fonte do sistema.
 - Este sistema entende-se como uma caixa, onde ao inserir valores de entrada (input), retorna valores de saída (output).
 - Dentro deste sistema os:
 - O teste de sistema são realizados por uma equipa específica de teste, que utiliza a especificação dada pelo cliente para fazer o percurso de casos de teste.
 - O teste de aceitação difere deste sistema, pois pode ser executado pelo cliente ou pelos utilizadores, cuja finalidade do teste é verificar se o sistema está de acordo com o que foi solicitado e se atende as necessidades dos utilizadores.



INTRODUÇÃO AOS TESTES APLICACIONAIS - TESTES NÃO-FUNCIONAIS

O QUE SÃO TESTES NÃO-FUNCIONAIS?

- Os testes não-funcionais, tratam-se de testes que verificam apenas aspectos gerais da aplicação, independente das regras de negócio que há nele. Como por exemplo, o teste ao desempenho, a segurança, à usabilidade, entre outros.



INTRODUÇÃO AOS TESTES APLICACIONAIS - TESTES DE REGRESSÃO

O QUE SÃO TESTES DE REGRESSÃO?

- Os testes de regressão devem ser realizados sempre que o sistema sofre alterações consideráveis, que pode gerar os chamados “bugs”.
- Geralmente é necessário re-executar todo o trajecto de teste criado para o teste funcional, desde que o sistema não seja muito grande.
- Para uma melhor eficiência nesse tipo de teste, é importante que ele seja automatizado, para ter melhores resultados e torná-lo viável.



INTRODUÇÃO AOS TESTES APLICACIONAIS – FASES DO PROCESSO

FASES DO PROCESSO DE TESTE?

Os testes devem ser executados de acordo com o apresentado de forma a torná-lo mais eficaz:

